

CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen VI-Fascículo 65

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones, Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, 08008 Barcelona
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-034-7 (tomo 6)
84-85822-82-X (obra completa)
Depósito Legal: B. 52/1984

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 108504

Impreso en España-Printed in Spain-Abril 1985

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 19 425 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 429 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S.A. (Paseo de Gracia, 88, 5.°, 08008 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.



Réplica perfecta

La mejor forma de "trabajar" con un robot en casa es mediante la creación de un modelo del mismo por ordenador

Steve Cross

Movimientos armónicos
Cuando los robots realizan una tarea conjunta existe una auténtica necesidad de coreografía, con el objeto de que no interfieran entre sí. Aquí, un brazo puede recoger y mantener el juguete en posición mientras el otro robot recoge un tambor y lo coloca en su sitio; el primer brazo coloca entonces el juguete ya completo en su caja. Si el movimiento del brazo se controla adecuadamente, las cintas transportadoras se pueden ubicar en cualquier disposición que resulte mejor para el resto de la cadena de montaje, mientras que las exigencias ergonómicas de un operador humano limitarían esta libertad

El desarrollo de la tecnología del ordenador ha llevado a una utilización cada vez mayor de las simulaciones: se pueden construir "modelos" por ordenador que imiten fidedignamente acontecimientos del mundo real. La mayoría de las personas ya están familiarizadas con el concepto de simuladores de vuelo, dispositivos sumamente complejos que permiten a los aspirantes a piloto adquirir experiencia de vuelo sin tener que pilotar un avión auténtico. Pero hay muchas otras actividades que también se prestan a la simulación por ordenador: previsiones comerciales, operaciones de ingeniería y procesos físicos de todo tipo se pueden simular muy fácilmente en un modelo por ordenador. En algunos casos, éste puede llevar a cabo experimentos que, de realizarse por algún otro medio, serían demasiado peligrosos. Por ejemplo, es de vital importancia descubrir lo que sucedería en una central de energía nuclear si se produjera una fuga de líquido refrigerante del reactor. En este caso, obviamente sería imposible utilizar una central nuclear auténtica para llevar a cabo el experimento, de modo que se emplea una simulación por ordenador. Si el modelo es suficientemente detallado, entonces se puede ver con exactitud lo que sucedería si se produjera la fuga.

Del mismo modo, en el campo de la robótica se utilizan simulaciones por ordenador para diseñar nuevos robots. Es evidente que se puede proceder mediante ensayo y error (construir un robot, observar cómo se comporta y luego efectuar todas las modificaciones necesarias), pero esto es costoso y lleva mucho tiempo. Una simulación por ordenador permite diseñar un robot y controlar sus acciones sin costo alguno y sin el esfuerzo físico que implica realizar frecuentes cambios de diseño.

Tomemos como ejemplo una cadena de montaje de automóviles, en la cual un equipo de robots va

trabajando en los coches a medida que van pasando. Todo lo que se desea es programar los robots de modo que ensamblen los vehículos de la forma correcta. No obstante, programar los robots ocupa tiempo y, cada vez que se paraliza la cadena de montaje, se pierde dinero. Podría optarse por crear una línea de montaje falsa con algunos robots totalmente diferentes con los cuales desarrollar los nuevos programas. Pero esto también es caro y puede conducir fácilmente a otra dificultad: el problema de la coreografía de los robots, de la cual ya hemos hablado anteriormente. Es de vital importancia asegurar que los robots que estén trabajando juntos no interfieran mutuamente en sus movimientos. Esto no es sólo una cuestión de conveniencia: un gran robot industrial capaz de transportar cargas pesadas podría con toda facilidad dañar a otro robot si tropezara con él. Y no sólo los robots pueden sufrir daños: ¡un robot programado incorrectamente que se pasara el tiempo soldando puertas de coches para cerrarlas enseguida sería desastroso!

Perfecta coordinación
La complejidad del movimiento y la sincronización que requieren las aplicaciones reales de la robótica se pueden apreciar claramente en esta sección de montaje del Ford Sierra

Cortesía de Ford

La respuesta obvia no es otra que llevar a cabo simulaciones por ordenador de las acciones de cada robot, de modo que el usuario pueda ver cómo interactúan las mismas. De esta forma los costos son bajos y no hay nada que pueda resultar dañado. Una vez que la simulación se haya completado y cuando todo parezca ser satisfactorio, se pueden transferir fácilmente los programas así desarrollados a robots reales, que entonces ya pueden llevar a cabo con total seguridad las tareas asignadas.

En este capítulo demostraremos el principio de la simulación por ordenador mediante un programa, *Brazo-robot*, que simula un brazo-robot de "coger y colocar" con dos grados de libertad. No posee sensores, de modo que usted mismo debe guiarlo, controlando las junturas del hombro y del codo y el mecanismo de prensión del efector final, con el fin de asir un objeto y colocarlo luego en algún otro lugar. Adicionalmente, remítase al programa para resolver laberintos que ofrecimos en la página 1202, que muestra cómo se puede programar un robot para que encuentre su camino hasta el centro de un laberinto. Este programa es, en realidad, una simulación por ordenador de cómo intentaría alcanzar su objetivo un robot real. Imita las acciones de un autómata cibernético dotado de un sencillo sensor táctil y encuentra la ruta correcta simplemente avanzando hacia espacios vacíos hasta hallar un callejón sin salida, punto en el cual retorna a la última intersección que había atravesado y trata entonces de seguir avanzando por un nuevo sendero. Este modelo no tiene prácticamente ninguna sofisticación, pero ilustra cómo se puede utilizar un programa de ordenador para simular los movimientos de un robot. El robot del programa obedece un conjunto estipulado de reglas y «traza un mapa» del entorno. Si, dentro del programa, tuviera acceso directo e inmediato a las posiciones de todos los componentes del laberinto, sería capaz de avanzar directamente hasta su meta. En nuestro programa el robot carece de esta información y, por tanto, debe valerse de una técnica de ensayo y error.

De forma similar, el programa *Brazo-robot* imita el comportamiento de un robot que no posee ningún sensor. Este programa contiene un modelo del entorno del robot y otro del brazo propiamente dicho, y es necesario asegurarse de que ambos modelos interactúan sólo como lo harían en la vida real. Por consiguiente, no se puede recoger un objeto con el brazo a menos que el mismo esté posicionado correctamente. Y tampoco es posible desplazar el brazo por debajo del nivel del suelo, dado que ello sería imposible si el brazo-robot fuera auténtico. A pesar de que estamos utilizando gráficos por ordenador, en los cuales una línea (que representa el brazo) podría fácilmente cruzarse con otra (el suelo), para una simulación exacta es preciso que estas líneas no se crucen. Y cuando el robot suelta un objeto, éste no debe permanecer necesariamente en esa posición: su simulación debe permitir efectos gravitacionales. Si este punto se ignorara, ¡ciertamente no se podría desarrollar una simulación segura para que un robot de "coger y colocar" manipulara huevos!

Para obtener mayor realismo

Existen muy pocas limitaciones en cuanto a lo que se puede conseguir utilizando la simulación por ordenador; y, en la mayoría de los casos, cuanto más compleja es la simulación, más fascinante resulta. Tal simulación puede ser aún más entretenida que el simple hecho de jugar con robots reales, por la sencilla razón de que, empleando una simulación, uno puede diseñar el robot que desee; la programación de los detalles correctos de éste y su mundo puede conducir a una mejor comprensión de los robots y de la forma que funciona el mundo físico. Observe otra vez el programa *Brazo-robot*. Verá que cuando el robot deja un objeto, éste cae al suelo y permanece allí. Para lograr que el modelo sea aún más realista, se puede modificar el programa de modo que el objeto vaya acelerando a medida que cae, obedeciendo, por lo tanto, la ley de gravedad. Y, tal vez, ¿podría incluso rebotar al caer al suelo?

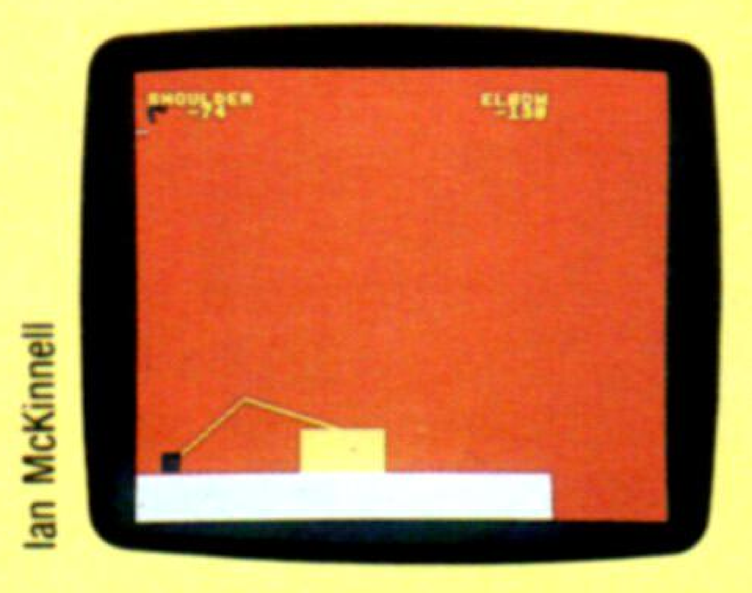
Las posibilidades son muchas y el programa está allí, para que usted lo adapte, añadiéndole características nuevas y más realistas para que su simulación sea lo más "viva" posible.

```
   4 REM ******SPECTRUM**********
   5 REM * SIMULACION BRAZO ROBOT *
   6 REM ******SPECTRUM**********
  10 CLS: PRINT "BRAZO ROBOT":PRINT:PRINT "LOS MANDOS PARA
     EL BRAZO ROBOT SON:"
  15 PRINT "S- SELECCION ROTACION HOMBRO":PRINT "E-
     SELECCION ROTACION CODO":PRINT "K- ROTACION JUNTURA
     SENTIDO HORARIO"
  20 PRINT "H- ROTACION JUNTURA SENTIDO ANTI-HORARIO":PRINT
     "U- COGER BOLA":PRINT "F- SOLTAR BOLA"
  25 PRINT AT 20, 11;FLASH 1; "pulse una tecla":PAUSE 0:
     RANDOMIZE
1000 GO SUB 9500:      REM inicio
1100 GO SUB 5000:      REM entrada
1200 GO SUB 6000:      REM caida
1400 STOP
1500 REM * actualizar junturaxy **********
1550 LET ex=l1*SIN hd1: LET ey=l1*COS hd1
1560 LET hx=sx+ex:LET hy=sy+ey
1650 LET wx=l2*SIN hd2: LET wy=l2*COS hd2
1660 LET hx=hx+wx:LET hy=hy+wy
1690 RETURN
2000 REM * dibujar brazo **********
2020 INK acol
2050 PLOT INVERSE rubout;sx,sy
2100 DRAW INVERSE rubout;ex,ey: DRAW INVERSE rubout;wx,wy
     : IF blup THEN LET br=FN r(hy): LET bc=FN c(hx):
     GO SUB 2500
2490 RETURN
2500 REM * dibujar bola **********
2600 INK bcol
2650 PRINT AT br,bc;"■ "(rubout+1)
2740 RETURN
2750 REM * soltar bola **********
2800 LET rubout=1: GO SUB 2000: LET rubout=0
2820 LET k=INT (xh*RND): IF k>=xs THEN IF k<=xs+wd THEN GO TO
     2820
2850 LET br=FN r(y0+4): LET bc=FN c(k): LET blup=0: GO SUB 2000:
     GO SUB 2500
2990 RETURN
3000 REM *       rotar       **********
3100 LET rubout=1: GO SUB 2000
3120 LET t1=dirn*sr*a1: LET t2=t1+dirn*er*a2
3150 LET hd1=hd1+t1: LET hd2=hd2+t2
3200 GO SUB 1500
3300 IF ABS hd1>p2 THEN LET ok=0
3320 LET pt=POINT (hx,hy)
3340 If pt<>0 THEN LET ok=0: IF br=FN r(hy) AND bc=FN c(hx) THEN
     LET ok=2
3400 LET rubout=0:GO SUB 2000
3450 INK bacol:PRINT AT 21,0;s$;AT 21,2;FN d(hd1);AT 21,26;
     FN d(hd2)
3490 RETURN
5000 REM *       entrada **********
5100 IF INKEY$<>" " THEN GO TO 5100
5120 FOR l=0 TO 1 STEP 0
5150 LET a$=INKEY$: IF a$>="A" AND a$<="Z" THEN LET
     a$=CHR$(CODE a$+32)
5200 IF a$="s" THEN LET sr=1:LET er=0
5220 IF a$="e" THEN LET er=1:LET sr=0
5250 IF a$="k" THEN LET dirn=1:GO SUB 3000
5270 IF a$="h" THEN LET dirn=-1:GO SUB 3000
5300 IF a$="u" THEN IF ok=2 THEN LET blup=1
5320 IF a$="f" THEN IF blup THEN GO SUB 2750
5400 IF NOT ok THEN LET l=2
5450 NEXT l
5490 RETURN
6000 REM *       caida **********
6100 PRINT AT 8,12;FLASH 1;"!!crash!!":BEEP .5,-5:BEEP
     1,-16:RETURN
9000 REM * dibujar base **********
9050 PAPER pacol:CLS
9100 INK grcol
9120 FOR k=0 TO y0:PLOT 0,k:DRAW xh,0:NEXT k
9200 INK bacol:LET xs=(xh-wd)/2
9220 FOR k=y0+1 TO y0+ht
9240 PLOT xs,k:DRAW wd,0
9260 NEXT k
9300 INK bcol:GO SUB 2500:INK acol
9400 PRINT AT 20,2;"HOMBRO";AT 20,26;"CODO"
9490 RETURN
9500 REM *       inicio ************
9550 DEF FN d(x)=INT (x*180/PI)
9560 DEF FN r(x)=21-INT(x/8)
9570 DEF FN c(x)=INT(x/8)
9600 DIM s$(32):LET xl=0:LET yl=0:LET xh=
     254:LET yh=174
9620 LET y0=23:LET wd=60:LET ht=23
9630 LET blup=0:LET bc=2:LET br=FN r(y0+4)
9640 LET grcol=3:LET bacol=2:LET acol=1:LET bcol=6:
     LET pacol=7
9650 LET sx=xh/2:LET sy=y0+ht+2:LET l1=(yh-ht-y0-2)/2:IF
     l1>xh/4 THEN LET l1=xh/4
9660 LET l2=l1:LET hx=0:LET hy=0
9670 LET p2=PI/2:LET a1=PI/32:LET a2=2*a1
9680 LET hd1=0:LET hd2=p2
9690 LET sr=1:LET er=0:LET dirn=1:LET rubout=
     0:LET ok=1
9750 GO SUB 9000:GO SUB 1500:GO SUB 2000
9790 RETURN
```

Coger y colocar

Este programa simula un brazo-robot que puede estirarse hasta alcanzar un objeto, asirlo y colocarlo luego en otro lugar. Su tarea consiste simplemente en recoger la bola y luego dejarla caer. El brazo está diseñado para utilizar coordenadas de revolución con dos grados de libertad: una juntura de hombro y una juntura de codo. La primera puede rotar a través de 180°; la segunda a través de 360°.

El programa se controla mediante las siguientes teclas: S indica que se desea un movimiento de hombro; E un movimiento de codo; las teclas K y H harán que la juntura del brazo rote en el sentido de las agujas del reloj o en el sentido contrario, respectivamente. Cada pulsación hace rotar la juntura del hombro en 6°, y en 12° la juntura del codo. U indica que se desea que el brazo intente asir la bola. Esto sólo se realizará con éxito si se ha logrado manipular el brazo dentro del alcance de la misma. F hará que el robot la deje caer





Ian McKinnell

Mover y recoger
Nuestro programa de simulación de un brazo-robot permite mover un brazo de dos junturas en dos dimensiones y recoger un objeto con el mismo. Cuando se deja caer el objeto, el programa lo coloca al azar sobre el suelo. La visualización muestra los ángulos verticales formados por los brazos superior e inferior

Para el BBC Micro realice las siguientes añadiduras y modificaciones:

```
   4 REM *****************BBC ****************
   5 REM*           SIMULACION BRAZO ROBOT         *
   6 REM *****************BBC ****************
   7 MODE 1:COLOUR 130:COLOUR 1:CLS
  25 PRINTTAB(15,20)"PULSE UNA TECLA":A$=GET$
1000 GOSUB 9600
2020 GCOL 0,acol
2050 MOVE sx,sy
2100 PLOT rubout,ex,ey:PLOT rubout,wx,wy:IF blup THEN
     br=hy:bc=hx:GOSUB 2500
2190 RETURN
2200 REM********** COGER LA BOLA **********
2250 blup=1:rubout=3:GOSUB 2500
2300 rubout=1:GOSUB 2000
2600 GCOL 0,bcol:MOVE bc, br
2650 PLOT 0,0,bsz:PLOT 80+rubout,bsz,0
2700 PLOT 0,0,-bsz:PLOT 80+rubout,-bsz,0
2800 rubout=3:GOSUB 2000:rubout=1
2820 k=INT(xh*RND(1)):IF k>=xs THEN IF k<=xs+wd THEN GOTO
     2820
2850 br=y0+5:bc=k:blup=0:GOSUB 2000:GOSUB 2500
3100 rubout=3:GOSUB 2000
3340 IF pt<>pacol-128 THEN ok=0:IF pt=bcol THEN ok=2
3400 rubout=1:GOSUB 2000
3450 COLOUR bacol:PRINTTAB(0,3)s$;TAB(4,3);FNd
     (hd1);TAB(27,3);FNd(hd2)
5100 IF INKEY$(0)<>" " THEN GOTO 5100
5150 a$=INKEY$(0):IF a$>="A"ANDa$<="Z"THENa$=
     CHR$(ASC(a$)+32)
5300 IF a$="u" AND ok=2 THEN GOSUB 2200
5400 IF ok=0 THEN l=2
6100 PRINTTAB(12,3)"!!CRASH!!":SOUND 1,-15,48,10:SOUND
     1,-15,4,20:RETURN
9050 GCOL 0,pacol:COLOUR pacol:CLS
9100 GCOL 0, grcol
9120 FOR k=0 TO y0:MOVE 0,k:DRAW xh,k:NEXT k
9200 GCOL 0,bacol:xs=(xh-wd)/2
9240 MOVE xs,k:DRAW xs+wd,k
9300 MOVE bc,br:GOSUB 2500:COLOUR acol
9400 PRINTTAB(2,2)"HOMBRO";TAB(26,2)"CODO"
9600 s$=" ":x1=0:y1=0:xh=1000:yh=1000
9620 y0=100:wd=200:ht=100
9630 blup=0:bsz=wd/5:bc=40:br=y0+5
9640 grcol=3:bacol=2:acol=2:bcol=0:pacol=129
9650 sx=xh/2:sy=y0+ht+2:l1=(yh-ht-y0-2)/2:IF l1>xh/4 THEN
     l1=xh/4
9690 sr=1:er=0:dirn=1:rubout=1:ok=1
```

Solución avanzada

"TK!Solver" aporta una nueva dimensión a la hoja electrónica: el proceso de ecuaciones

Como ya hemos visto en este apartado, los programas de hoja electrónica para microordenadores pueden ser muy útiles para una gran variedad de tareas matemáticas. Para la persona acostumbrada a trabajar con grandes hojas compuestas de filas y columnas, con un lápiz y una calculadora, la hoja electrónica representa un valioso elemento que ayuda a ahorrar tiempo y esfuerzo. No obstante, las hojas electrónicas poseen limitaciones significativas. El formato de fila y columna, ideal para contabilidad u otros modelos financieros, a menudo es incómodo y, en ocasiones, inútil para aplicaciones científicas y matemáticas de nivel más elevado. Y las hojas electrónicas tienen una estructura muy rígida para manipular ecuaciones.

Software Arts, la empresa norteamericana que creó el *VisiCalc*, ha desarrollado un programa llamado *TK!Solver*, que va más allá de las hojas electrónicas tanto en su forma como en sus funciones. "*TK!*" alude a *ToolKit* (juego de herramientas), mientras que "*Solver*" (el que soluciona) es la sección de código que procesa ecuaciones. Además de ser diferente a las hojas electrónicas desde el punto de vista del formato de la pantalla, *TK!* ofrece las siguientes características exclusivas:

Modelo de interacción

Este diagrama refleja la estructura de las hojas de trabajo disponibles en *TK!Solver* y la relación que guardan entre sí. Las subhojas contienen datos a los que se llama desde la hoja controladora

Liz Dixon

Backsolving: Las fórmulas de hoja electrónica sólo pueden resolver una única variable. *TK!* puede resolver cualquier variable de una ecuación si se le proporcionan datos suficientes para hacerlo;
Iteración: Si falta o no se conoce algún valor necesario para resolver una ecuación, se puede entrar un valor de ensayo y *TK!* lo utilizará como punto de partida. Luego resolverá la ecuación a través de una serie de aproximaciones sucesivas;
Conversión de unidades: Puede convertir valores de pies a metros, de dólares a libras, etc., instantáneamente a partir de tablas de conversión;
Funciones matemáticas: Tiene incorporadas una enorme variedad de ellas.

Hojas de trabajo

TK!Solver opera a través de tres hojas de trabajo enlazadas, cada una de ellas con una función específica. La hoja Variable contiene los nombres de todas las variables definidas, columnas para los valores que entra el usuario y los valores que produce el programa, un lugar para indicar unidades que guardan alguna relación, y un espacio para que el usuario anote un comentario para cada variable. La hoja Variable aparece en la parte superior de la pantalla que visualiza inicialmente el programa. Cada variable también se escribe en detalle en una subhoja de variables separada. La hoja Rule se utiliza para entrar las ecuaciones que *TK!* ha de resolver. Una ecuación puede tener hasta 200 caracteres de longitud, y debe responder a las convenciones matemáticas estándares en cuanto a notación y operaciones. La hoja Rule ocupa la porción inferior de la pantalla inicial del programa. La hoja Unit almacena la información necesaria para convertir las unidades de medida que acompañan a las variables de un modelo.

TK! emplea estas tres hojas para llevar a cabo la mayoría de sus operaciones. Entre las otras se incluyen la hoja Global, en la que el usuario puede construir algunos de los procedimientos operativos del programa; la hoja List, que almacena una matriz de valores para variables; la hoja User Function, para funciones definidas por el usuario; y hojas para trazar o imprimir puntos o tablas de valores.

Creación de un modelo

Empezaremos por crear un modelo muy simple adaptado del manual de usuario del *TK!*, que calcula los gastos del recorrido y la velocidad media para un viaje en automóvil, y convierte los valores de unidades inglesas a unidades métricas. En la visualización inicial encontramos el cursor en la hoja Rule, en la parte inferior de la pantalla. Comenza-

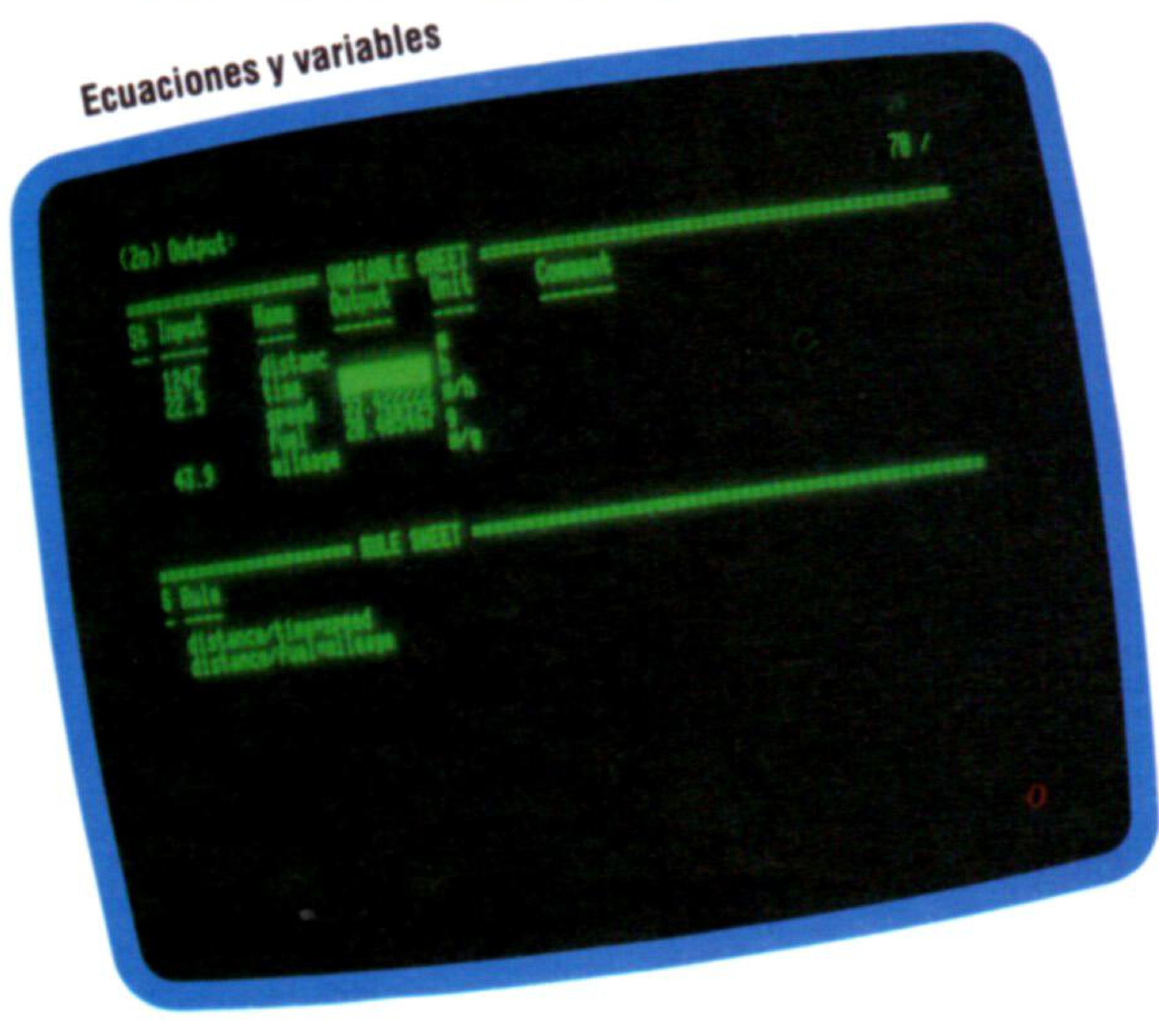
mos por definir las variables en ecuaciones apropiadas, de modo que tecleamos:

distancia/tiempo=velocidad

y pulsamos Return. Inicialmente, el programa está preparado para leer nombres de variables de ecuaciones directamente en la hoja Variable de la parte superior. *TK!* evalúa la ecuación y visualiza las variables en la columna Name de la hoja Variable por el mismo orden en que aparecen en la ecuación. Luego se visualiza un asterisco en la columna Status (estado) al lado de la ecuación. El asterisco significa que en la hoja de variables no se ha entrado ningún valor. Luego entramos la segunda ecuación por el mismo procedimiento:

distancia/gasolina=gastosviaje

Ecuaciones y variables



Después de entrar esta ecuación, las cinco variables definidas se listarán en la columna Name de la hoja Variable.

Pulse la tecla del punto y coma (;) para mover el cursor desde la hoja Rule hasta la hoja Variable, en la cual ya podemos entrar valores. El cursor aparece en la columna de entradas junto a nuestra primera variable, distancia. Se entran los siguientes valores digitándolos en el espacio apropiado, pulsando luego Return o la flecha del cursor que señala hacia abajo.

INPUT (entrada)	**NAME** (nombre)	**OUTPUT** (salida)
500	distancia	
8.5	tiempo	
	velocidad	
14	gasolina	
	gastosviaje	

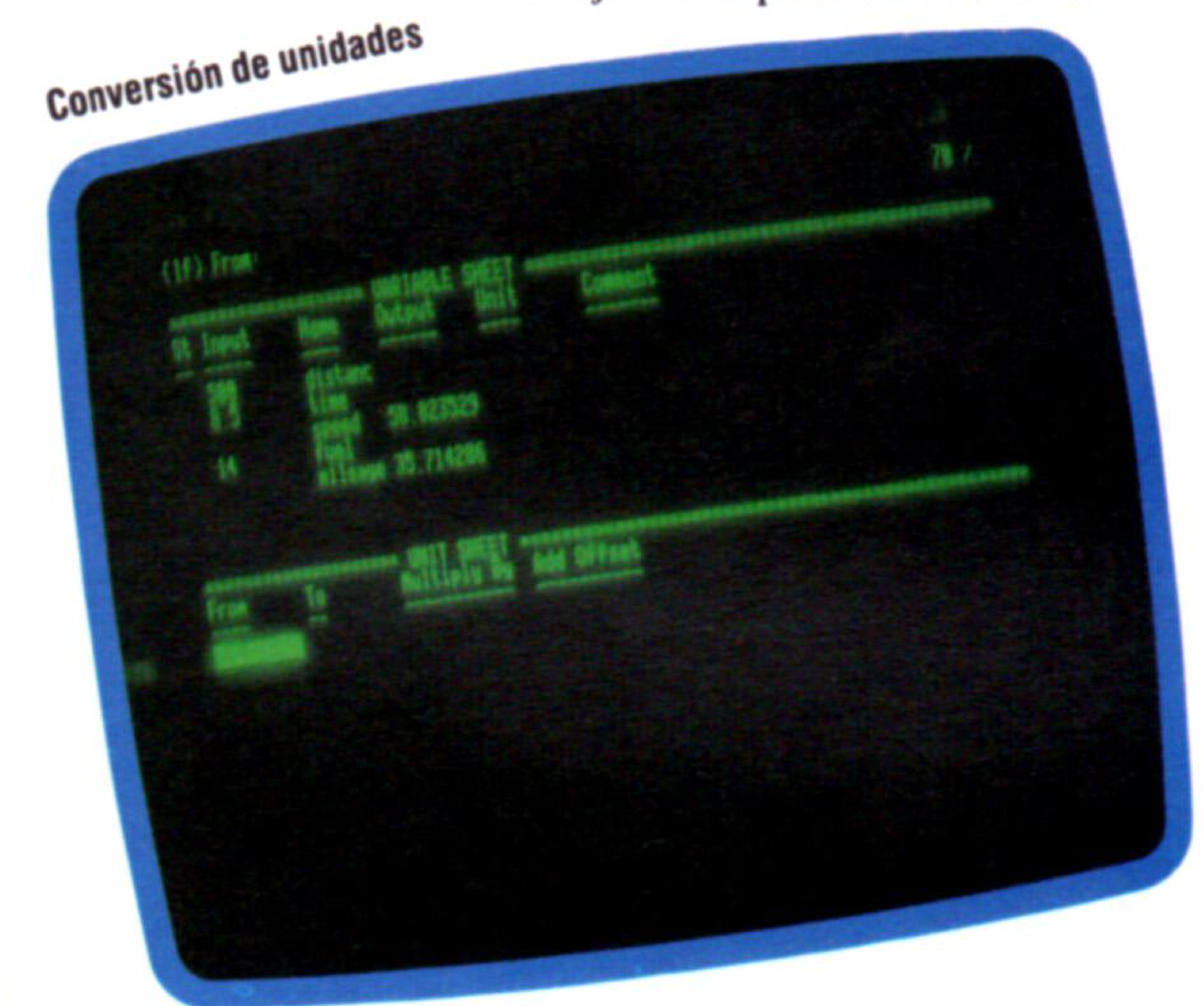
Los valores de velocidad y gastosviaje se dejan en blanco para que los resuelva *TK!* Sus valores calculados se visualizarán en la columna OUTPUT. Para resolver velocidad y gastosviaje, pulse el signo de exclamación (!). Se visualizará la frase Direct Solver (solución directa) en la parte superior de la hoja Variable, porque al programa se le han proporcionado todos los datos necesarios para hallar una solución directa. Enseguida se visualizarán como salida los valores para velocidad y gastosviaje. Podemos eliminar los valores entrados previamente y obtener una cifra para distancia dándole a *TK!* nuevos valores para velocidad y tiempo, o para gastos del viaje y gasolina.

Hasta ahora los valores entrados en nuestro modelo no llevan asociada ninguna unidad. No podemos digitar millas ni galones en la columna de unidades de la hoja Variable, porque no se puede utilizar unidades hasta no haberlas definido. Desplazamos el cursor a la hoja Rule pulsando la tecla del punto y coma (;) y digitando luego =U. *TK!* reemplaza la hoja Rule de la ventana inferior por la hoja Unit. Ésta posee cuatro columnas:

From (de)	To (a)	Multiply by (multiplicar por)	Add offset (añadir resto)

El cursor se visualiza debajo de la palabra From. Ya

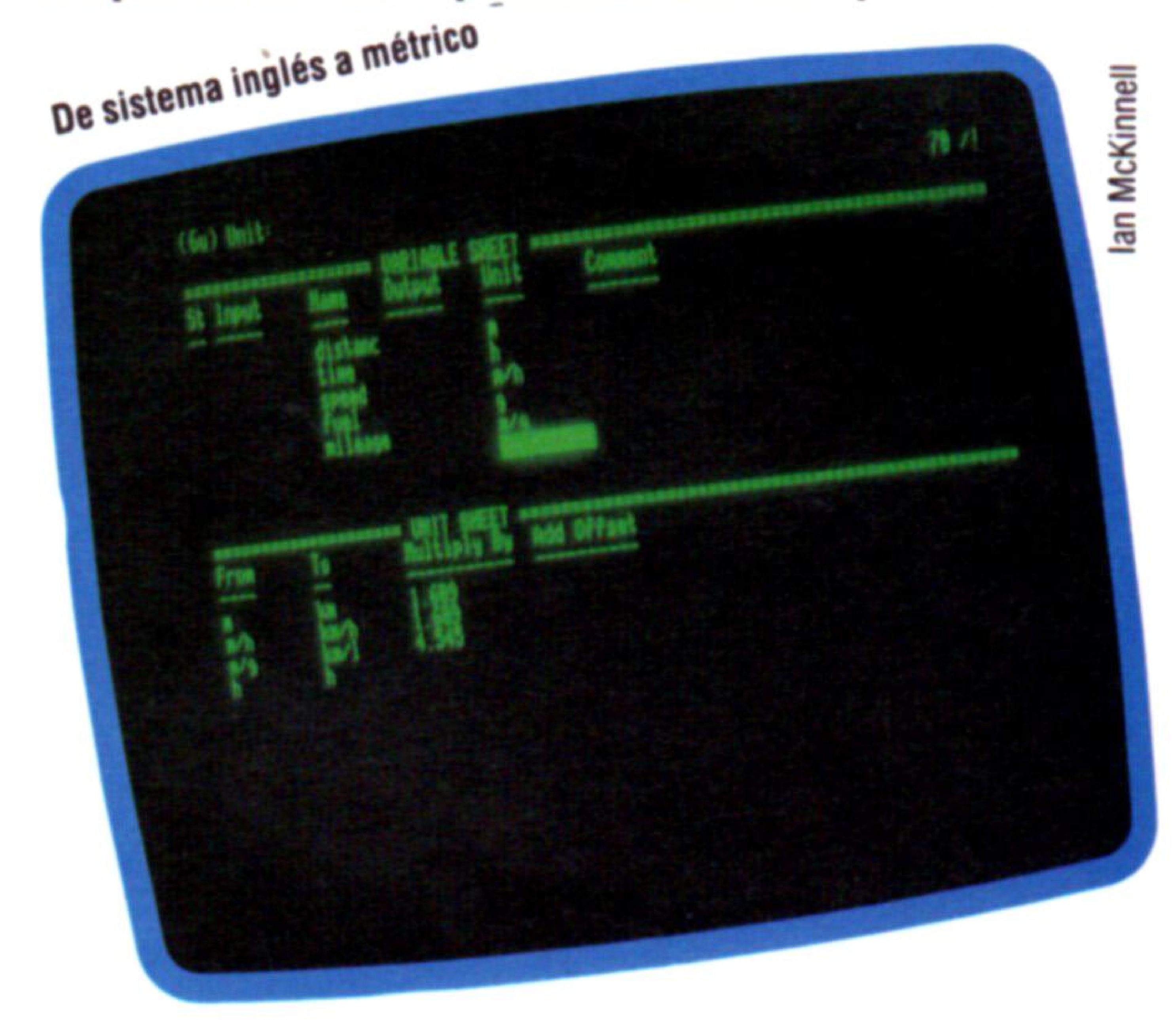
Conversión de unidades



podemos entrar las unidades que deseamos que *TK!* conozca y los valores de conversión.

Después pulse =R para recuperar la hoja Rule; seguido de ; para pasar a la hoja Variable. Ahora ya es posible entrar los nombres definidos en la columna Unit: m para distancia; h para tiempo; m/h para velocidad; g para gasolina y m/g para gastosviaje. Vacíe todos los valores en curso y sustitúyalos por éstos: 1,247 para distancia; 22.5 para tiempo

De sistema inglés a métrico



Ian McKinnell

y 43.9 para gastosviaje. Pulse ! para resolver y se visualizarán los valores métricos.

Ahora coloque el cursor sobre m, en la columna de unidades, y digite km de kilómetros. Pulse Return, y *TK!* convertirá automáticamente el valor de 1,247 para distancia de millas a kilómetros, de modo que el valor para distancia se cambia por 2006.423.

En este simple modelo hemos utilizado sólo algunas de las facilidades que ofrece *TK!*

En la línea del QL

Como respuesta a la aparición de máquinas más avanzadas, Sinclair ha vestido al Spectrum con nuevos ropajes

En la época de su lanzamiento, en la primavera de 1982, el Sinclair Spectrum ofrecía una notable relación calidad/prestaciones. Sus únicos competidores auténticos fueron el Vic-20, con unos escasos 3,5 K de memoria para el usuario, y el Texas TI99 4A, que se vendían al doble de su precio. De este modo el Spectrum se convirtió en un éxito instantáneo entre quienes adquirían un ordenador personal por primera vez, además de convertirse en la elección natural de los millares de entusiastas de los micros que desearon mejorar sus ZX80 y ZX81. La nueva máquina tenía una memoria asombrosa de 48 K, utilizaba un buen BASIC y ofrecía ocho colores para gráficos, así como una primitiva facilidad de sonido. El teclado, asimismo, representaba una gran mejora respecto a la lámina plástica plana "sensible al tacto" del ZX81. A la venta inicialmente sólo por correspondencia, el Spectrum fue un éxito inmediato y enseguida se convirtió en el micro más vendido.

En el tiempo transcurrido desde su lanzamiento, los competidores de Sinclair han producido una gama de máquinas para desafiar el dominio del mercado ejercido por el Spectrum. A pesar de tener un BASIC claramente inferior, el Commodore 64 fue el competidor que más éxito obtuvo; ofrecía más memoria (si bien necesitaba programas en lenguaje máquina para aprovecharla al máximo), un sonido excelente y un "verdadero" teclado, con teclas móviles tipo máquina de escribir. Asimismo, el BBC Micro ofrecía mejores especificaciones, pero su precio impidió que se convirtiera en una seria amenaza, y sus fabricantes optaron por no reducir su precio. Sin embargo, sucesivos recortes en los precios del Commodore redujeron sustancialmente el costo del 64. La reacción de Sinclair fue rebajar el precio del Spectrum.

Para entonces, el teclado del Spectrum (que en su tiempo fuera motivo de atracción) era ya un verdadero inconveniente. La base de software de la máquina no había sido superada y se produjeron muchos paquetes "serios" para ella. No obstante, tratar de utilizar programas para tratamiento de textos con el teclado del Spectrum era como escribir a máquina con un par de mitones puestos. Muchos usuarios, por consiguiente, invirtieron en teclados "adecuados" y esta tendencia se acentuó al aparecer la unidad Interface 1/Microdrive, tanto tiempo esperada. Enseguida se hizo evidente que los usuarios de micros de 1984 ya no estaban preparados para aceptar la idea de Sinclair de lo que constituía un dispositivo de entrada aceptable.

Seis en uno
Haciéndose eco de la moda actual de "empaquetar" software con los micros personales, Sinclair incluye con el Spectrum+ (y con el Spectrum de 48 K) un impresionante paquete de software compuesto por seis programas. Comprende un procesador de textos, una hoja electrónica, dos juegos y dos paquetes para gráficos



Cirugía plástica

La respuesta de Sinclair Research ha sido practicarle al Spectrum una cirugía plástica facial. El Spectrum+ es esencialmente la misma máquina, pero alojada en un teclado tipo QL, si bien reducido, con unas pocas teclas extras, un interruptor de Reset y un par de patas retráctiles. Todos los periféricos producidos para la versión más antigua trabajarán con el "Plus" (más), pero Sinclair no ha sabido aprovechar la oportunidad para colocar el rendimiento del Sinclair más en la línea de las máquinas rivales, mejorando el sonido o proporcionando un conector para monitor o una Interface 1 incorporada. Ahora las facilidades de sonido de la máquina constituyen su punto más débil. Las dos patas permiten que se escape un poco más de volumen de la base de la máquina, pero esto es más una incomodidad que una ventaja, puesto que significa que también se magnifican los ruidos propios de cargar (LOAD) y guardar (SAVE). El patético BEEP del Spectrum sigue siendo tristemente inadecuado.

El Spectrum+ mide 319 ×149 ×38 mm. El nuevo diseño hace más sencilla la programación al proporcionar teclas extras para las modalidades "Extended" y de gráficos, video normal e invertido, teclas Delete y Break y teclas separadas para los signos de puntuación que se utilizan comúnmente, como el punto y coma, las comillas, la coma y el punto. También se ha agregado una tecla Symbol Shift extra, y las teclas del cursor están ahora colocadas en lugares nuevos, a lo largo de una pequeña barra espaciadora. Todas las combinaciones de teclas antiguas siguen funcionando. A los antiguos usuarios del Spectrum el nuevo diseño podría causarles algunos problemas. En especial, la nueva tecla Edit está situada junto a la tecla "A"; si ésta se pulsa por error cuando se está entrando una línea de programa, la línea se pierde.



La placa de circuitos

En el interior de la atractiva carcasa tipo QL del Spectrum+ encontramos la placa de circuitos Issue 4.5. Ésta es esencialmente la misma que la placa Issue 3 del Spectrum de 48 K que salió al mercado en agosto de 1983, con la adición de los desafortunados cables volantes del botón de reset parcheando la placa

Chris Stevens

SPECTRUM+

DIMENSIONES

319 × 149× 38 mm

MEMORIA E INTERFACES

Similares a las del Spectrum de 48 K, BASIC residente compatibilidad total de software

TECLADO

58 teclas esculpidas (incluyendo una barra espaciadora); teclado tipo membrana

DOCUMENTACION

Manual ilustrado en color con cassette de guía para el usuario

VENTAJAS

La riqueza del software existente para el Spectrum, los grupos de usuarios y las publicaciones especializadas constituyen envidiables atractivos

DESVENTAJAS

A pesar de las teclas nuevas y de las patas, el "tacto" y la acción del teclado continúan siendo un punto débil

Como parte del reempaquetamiento, los compradores del Sinclair reciben un lote compuesto por seis programas: *Psion chess, Make-a-chip, Scrabble, Chequered flag, Vu-3D* y el excelente software para tratamiento de textos *Tasword two*. Todos estos programas responden a un estándar muy elevado. Lamentablemente, no se puede decir lo mismo de la nueva documentación del Spectrum, la cual, a pesar de estar cuidadosamente presentada, carece de la profundidad del antiguo manual del Spectrum. Los editores sugieren, no obstante, que a medida que los usuarios adquieran mayor experiencia con la máquina, pueden hacerles llegar manuales más exhaustivos.

Para juegos, la nueva versión es ciertamente mejor que la original, pero los usuarios más entusiastas es posible que ya hayan invertido en una palanca de mando y una interface. Tanto la interface Kempston como la Fuller funcionan con el Spectrum+, si bien, como sucede con la máquina más antigua, el empleo de la Fuller Soundbox puede hacer inviable la ejecución de cierto software. La interface Centronics de Kempston también funciona a la perfección, al igual que el sistema Wafadrive de almacenamiento masivo.

El Spectrum+ representa, por cierto, una mejora sobre el Spectrum original, pero la idea que tiene Sinclair sobre lo que es un teclado razonable no va a contar con una aprobación universal. Si bien por parte de Sinclair se puede considerar una jugada inteligente la utilización de la tecnología del QL en un esfuerzo por hacer que el Spectrum resulte más atractivo a los compradores, hay que comparar el aumento en el precio del nuevo teclado con los accesorios ya disponibles. Bajo este prisma, no se puede considerar que la relación calidad/precio sea muy buena. Ciertamente, la nueva máquina tiene un aspecto más elegante, pero las teclas, a pesar de que están "esculpidas" para facilitar el tecleado, carecen de capacidad de respuesta y están demasiado apiñadas.

Para quien adquiera su primer ordenador, el Spectrum+ representa una opción que merece ser considerada, pero las facilidades que ofrece no compensan el aumento en su precio. Algunos usuarios podrían aventurar que Sinclair ha introducido el Spectrum+ simplemente como una forma de aumentar los precios; no sería sorprendente que el modelo original se retirara pronto de la venta. De hecho, la introducción de este modelo es un gesto extrañamente desinteresado por parte de Sinclair: seguramente habría sido más lógico rebajar el precio de la versión antigua (y del paquete Interface 1/Microdrive) y dejarlo así. Por otra parte, Sinclair podría haber aumentado ligeramente el precio e incluido todas las cosas que el Spectrum necesita *verdaderamente*, como unas adecuadas facilidades de sonido, un teclado de teclas móviles, conector para monitor e, incluso, un microdrive incorporado. Pero, de haber sido así, jamás hubiera estado listo a tiempo para las Navidades de 1984.

El teclado

Pensar que el teclado estilo QL del Spectrum+ representa una mejora o no, depende mucho del estilo de mecanografía que usted posea; la provisión de teclas separadas para las funciones más importantes sí representa una positiva innovación. Las secuencias de teclas originales del Spectrum (p. ej., [SYM SHIFT]+[0] para el punto y coma) duplican los efectos de las teclas nuevas

Ian McKinnell

Final del juego

Con la aparición de un fabuloso genio llega a su fin nuestro juego de aventuras en LOGO

Nuestro juego *El santuario de Zoltoth* sólo tiene incorporados dos "peligros". En CUARTO.4, el jugador se enfrenta a una serpiente poco amistosa, y el programa bifurca a un procedimiento de "peligro" especial:

```
TO ATAQUES.SERPIENTE
  PRINTL [[HAY UNA ENORME SERPIENTE] [QUE
  AVANZA LENTAMENTE HACIA TI!]]
END
```

El otro "peligro" no coloca al jugador en ninguna situación de peligro físico inmediato, pero, por cierto, podría causar problemas a largo plazo:

```
TO PUERTA
  PRINTL [[UNA GRAN PUERTA DORADA SE
  CIERRA DETRAS DE TI] [CERRANDOTE LA
  SALIDA POR EL SUR]]
  MAKE "PELIGROS []
  MAKE "LISTA.SALIDAS [[N 7] [E 8]]
END
```

Hay otras consideraciones que en ciertos puntos del programa es necesario tener en cuenta. El procedimiento COGER debe modificarse para que no se pueda recoger el anillo si se está llevando la espada.

```
TO COGERLO :ITEM
  IF :ITEM="ANILLO THEN COGER.ANILLO
  STOP
  AGREGAR.A.INV :ITEM
  SACAR.DEL.CUARTO :ITEM
END

TO COGER.ANILLO
  IF MEMBER? "ESPADA :INVENTARIO THEN PRINT
  [NO PUEDES RECOGER EL ANILLO] STOP
  AGREGAR.A.INV :ITEM
  SACAR.DEL.CUARTO :ITEM
END
```

Ésta es la única restricción en cuanto a que el jugador coja un objeto. Las siguientes rutinas permiten examinar lo que sea que esté sosteniendo.

```
TO EXAMINAR :OBJ
  IF :OBJ= "ANILLO THEN DESC.ANILLO
  STOP
  IF :OBJ= "COFRE THEN DESC.COFRE
  STOP
  IF :OBJ= "ESPADA THEN DESC.ESPADA
  STOP
  PRINT [NO LLEVAS NADA ESPECIAL]
END

TO DESC.ANILLO
  IF AQUI? "ANILLO THEN PRINTL [[EN EL ANILLO
  HAY UNA INSCRIPCION BORROSA:]
  [R-- -E]] ELSE PRINT [NO VEO NINGUN
  ANILLO]
END

TO AQUI? :OBJ
  IF MEMBER? :OBJ :CONTENIDO THEN OUTPUT
  "TRUE IF MEMBER? :OBJ :INVENTARIO THEN
  OUTPUT "TRUE OUTPUT "FALSE
END

TO DESC.COFRE
  PRINTL [[ES MUY HERMOSO] [Y
  EVIDENTEMENTE VALE UNA PEQUEÑA
  FORTUNA] [HAY UNA PEQUEÑA CALAVERA
  TALLADA] [EN UNA ESQUINA DE LA TAPA]]
END

TO DESC.ESPADA
  IF AQUI? "ESPADA THEN PRINT [ES DE ACERO]
  ELSE PRINT [NO VEO NINGUNA ESPADA]
END
```

El jugador necesita la espada para matar a la serpiente; si no la tiene, entonces es el ofidio el que lo mata a él.

```
TO MATAR :LA
  IF :LA="SERPIENTE THEN MATAR. SERPIENTE
  STOP
  PRINT [NO PUEDES HACER ESO!]
END

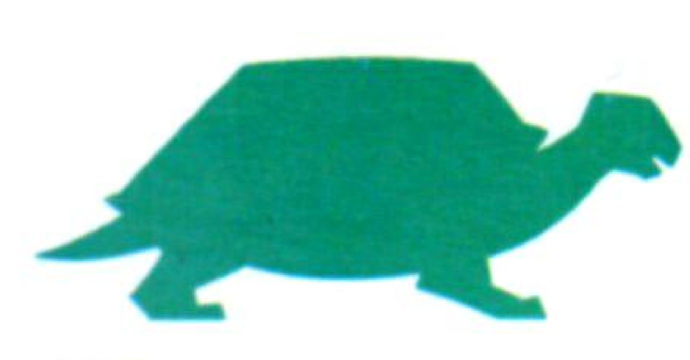
TO MATAR.SERPIENTE
  IF NOT MEMBER? "ATAQUES.SERPIENTE
  :PELIGROS THEN PRINT [NO VEO NINGUNA
  SERPIENTE] STOP
  IF MEMBER? "ESPADA :INVENTARIO THEN
  SERPIENTE.MUERE ELSE SERPIENTE.MATA
END

TO SERPIENTE.MUERE
  PRINT [LA SERPIENTE MUERE, ENROLLANDOSE
  EN SU AGONIA]
  MAKE "PELIGROS []
END

TO SERPIENTE.MATA
  PRINTL [[TU NO TIENES NINGUN ARMA] [CON LA
  CUAL MATARLA] [PERO AHORA YA ESTA
  TOTALMENTE ENLOQUECIDA] [TE MUERDE! TU
  ROSTRO SE ENSOMBRECE] [Y CAES AL SUELO
  RETORCIENDOTE]]
  MUERTO
END
```

El procedimiento MUERTO se anticipa sagazmente a aquellos jugadores que crean en la reencarnación. Si, después de haber muerto, usted digita cualquier otra cosa que no sea EMPEZAR, ¡el ordenador le recordará que está muerto!

```
TO MUERTO
  PRINT [ESTAS MUERTO!]
  PRINT1 "?
  MAKE "INPUT INSTRUCCION
  IF (:INPUT ="EMPEZAR)THEN EMPEZAR STOP
```





```
    PRINT [NO ME VENGAS CON ESO!]
    MUERTO
END

TO INSTRUCCION
    MAKE "INP REQUEST
    IF :INP=[] THEN PRINT1 "? OUTPUT
    INSTRUCCION
    OUTPUT FIRST :INP
END
```

Al frotar el anillo, ante los ojos del usuario hace su aparición un genio:

```
TO FROTAR :OBJ
    IF :OBJ= "ANILLO THEN FROTAR.ANILLO STOP
    PRINT [AHORA ESTA MUCHO MAS LIMPIO QUE
    ANTES]
END

TO FROTAR.ANILLO
    IF AQUI! "ANILLO THEN GENIO ELSE PRINT [NO
    VEO NINGUN ANILLO]
END
```

El genio le ofrece llevarlo a casa, pero si rechaza la invitación, entonces un fuerte viento lo llevará al azar hasta un cuarto situado en la parte oriental de la cueva:

```
TO GENIO
    PRINTL [[APARECE UN GENIO Y TE PREGUNTA:]
    ["QUIERES QUE TE LLEVE DE REGRESO A
    CASA?"]]
    PRINT1 "?
    MAKE "RESP FIRST INSTRUCCION
    IF ANYOF:RESP="SI :RESP="S THEN
    REGRESAR ELSE SOPLAR
END

TO REGRESAR
    PRINT [AL FIN EN CASA]
    IF MEMBER? "CETRO :INVENTARIO THEN PRINT
    [FELICITACIONES HAS ENCONTRADO EL CETRO!]
    ELSE PRINTL [[BUENO AL MENOS HAS
    ESCAPADO] [VIVO]]
END

TO SOPLAR
    PRINT [SOPLA UN FORTISIMO VIENTO]
    PRINT "
    MOVER1 (6+(RANDOM 5))
END
```

Mordedura mortal

Lo único que se puede abrir es el cofre, y éste contiene una araña venenosa. La calavera de la tapa es una advertencia para no abrirlo; pero, ya se sabe, ¡algunas personas nunca aprenden!, y las consecuencias, en este caso, son de imaginar.

```
TO ABRIR :OBJ
    IF :OBJ="COFRE THEN ABRIR. COFRE ELSE
    PRINT [NO LO PUEDES ABRIR]
END

TO ABRIR.COFRE
    PRINTL [[HAY UNA ARAÑA VENENOSA] [EN EL
    COFRE] [TE MUERDE]]
    MUERTO
END
```

Por último, he aquí un listado de todos los sustantivos del juego:

```
TO ESPADA
    OUTPUT "ESPADA
END

TO COFRE
    OUTPUT "COFRE
END

TO CETRO
    OUTPUT "CETRO
END

TO ANILLO
    OUPTUT "ANILLO
END

TO SERPIENTE
    OUTPUT "SERPIENTE
END
```

Si desea conservar el estado del juego para continuar en otro momento, simplemente digite SAVE "AVENTURA, y se guardará todo el contenido del espacio de trabajo. Se recuperará mediante READ "AVENTURA.

En muchos sentidos, el LOGO es un lenguaje ideal para programar juegos de aventuras. Existe, no obstante, un problema: en las implementaciones del lenguaje que existen hoy en día, no hay suficiente espacio. El juego que ofrecemos aquí apenas si cabe en la asignación de memoria para el Commodore 64. Cualquier ampliación que se introduzca en nuestro juego de aventuras en LOGO exigirá que el usuario adopte una decisión respecto a qué palabras conservar y cuáles suprimir.

Complementos al LOGO

Algunas versiones de LOGO MIT no poseen EMPTY?, ITEM, COUNT ni MEMBER? Las definiciones para las mismas ya las hemos ofrecido. En todas las versiones LCSI, utilice

EMPTYP por EMPTY?
LISTP por LIST?
MEMBERP por MEMBER?
TYPE por PRINT1
AND por ALLOF
OR por ANYOF

Hay una primitiva, EQUALP, que comprueba si sus dos entradas son la misma. Utilícela para comparar listas y palabras en lugar del signo de igualdad (que funciona para listas sólo en algunas versiones LCSI). La sintaxis IF en el LOGO LCSI se demuestra mediante:

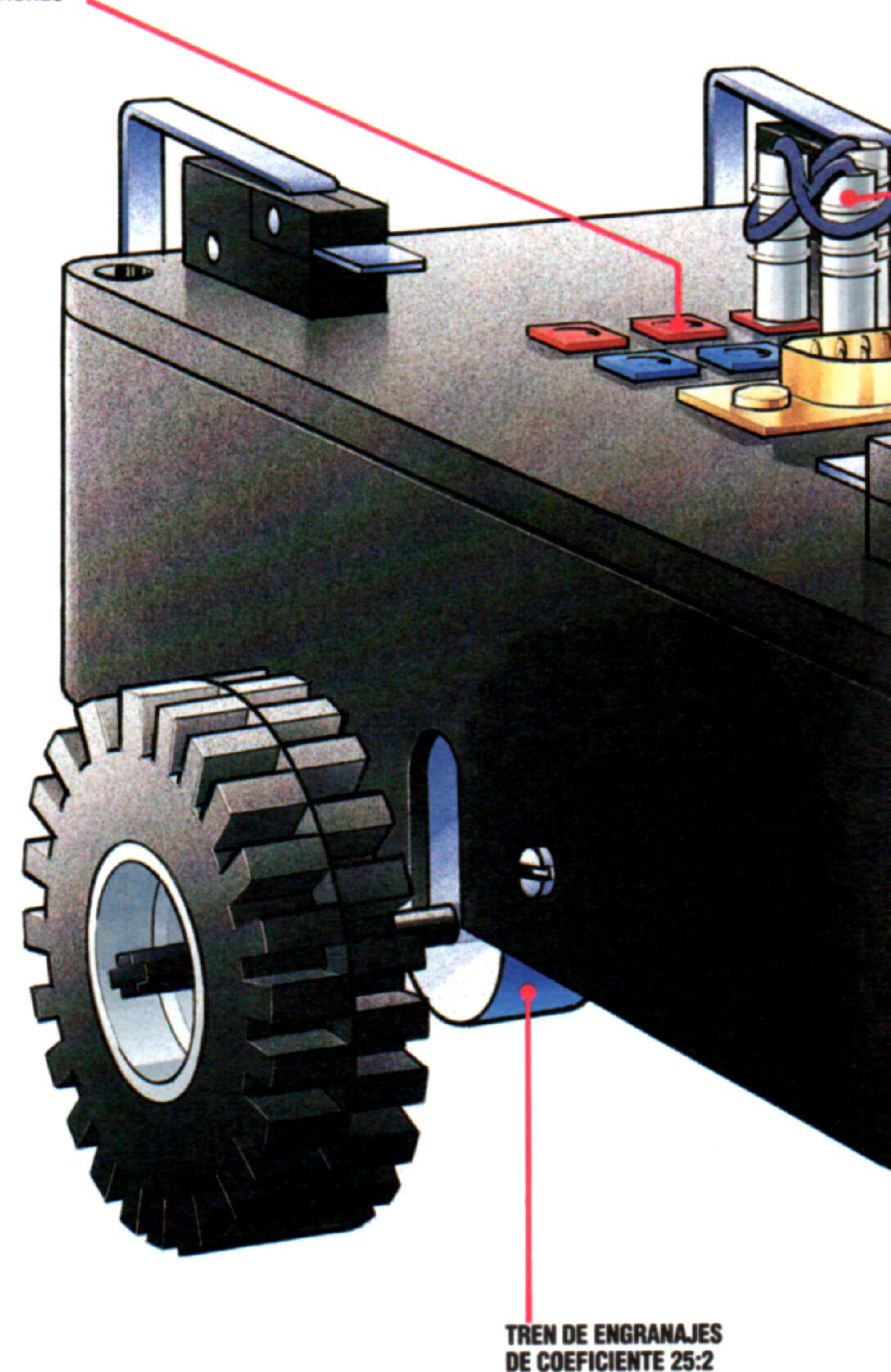
```
IF EMPTYP :CONTENIDO[PRINT [NADA
ESPECIAL]] [PRINT :CONTENIDO]
```

Si la condición es verdadera se lleva a cabo la primera lista después de la condición; y, si es falsa, se lleva a cabo la segunda.
En el LOGO Atari utilice SE por SENTENCE, RL por REQUEST, y observe que ITEM no está implementada.
La versión que se ofrece en el texto se ejecutó en el Commodore 64; puede ser que otras máquinas no posean espacio suficiente para ejecutar todo el juego tal como está. Si fuera éste el caso, tendrá que reducir el tamaño del juego, omitiendo algunas palabras descriptivas.

Liz Dixon

En el principio

Iniciamos un nuevo proyecto: la construcción de un robot con ruedas y sensores de proximidad y de luz



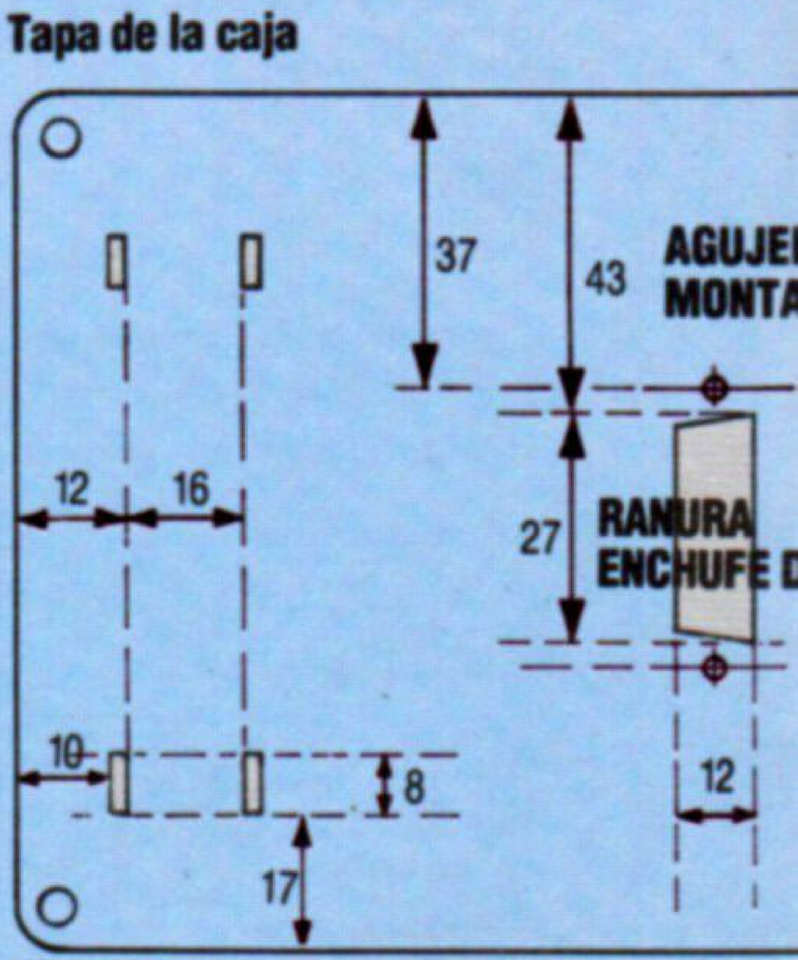
El robot estará alimentado por dos motores paso a paso, que accionarán dos ruedas mediante un sistema de engranaje. Los motores paso a paso que utilizaremos se pueden controlar para girar en pasos discretos de 7,5°. Accionar el motor a través de una caja de engranajes de un coeficiente de 25:2 significa que las ruedas del vehículo se podrán controlar con total precisión para una rotación del eje de 0,6°. Dado que los motores paso a paso trabajan girando a través de un ángulo discreto cada vez que reciben un impulso, resultan ideales para el control a través de un dispositivo digital. Emplearemos la puerta para el usuario del ordenador como nuestra fuente de control digital, lo que nos permitirá diseñar un software sencillo para usar conjuntamente con el robot. Además de estar equipado con motores paso a paso, el robot ya acabado contará con una gama de sensores, incluyendo sensores de proximidad y un par de sensores luminosos para permitir que siga una línea trazada en el suelo. Dado que se requieren cuatro líneas de datos de la puerta para el usuario para controlar los motores del vehículo, sólo quedan disponibles otras cuatro líneas para las entradas provenientes de los sensores. Para conseguir la máxima flexibilidad, el robot estará dotado de un sistema de "parches". Esto significa que a las cuatro líneas de datos disponibles se pueden conectar distintas combinaciones de sensores mediante algunos conectores montados en el robot y el empleo de cortos trozos de cable. Por ejemplo, puede que una aplicación requiera los cuatro sensores de proximidad, mientras que otra necesite dos sensores de proximidad y dos sensores luminosos. Con el sistema de parches los sensores requeridos se pueden enchufar directamente en las líneas de entrada de datos correspondientes.

Al poder controlar el robot con toda precisión, y dado que el mismo estará dotado de sensores, asumiremos igualmente la tarea de diseñar algo de software refinado para posibilitar la creación de un mapa interno del entorno inmediato del robot. Entonces podremos comenzar a investigar los detalles propios de los algoritmos de planificación de ruta y estrategia de búsqueda. En este primer capítulo comenzamos con la construcción mecánica del robot. Ésta es bastante directa e implica perforar y cortar la caja plástica que forma la carcasa y el chasis del robot; el posicionamiento del tren de engranajes y de los agujeros para el montaje de los enchufes debe ser exacto.

Primer paso

En primer lugar corte los agujeros necesarios en la carcasa plástica que ha de alojar y conformar el chasis del robot terminado. El diagrama ilustra la posición y el tamaño de los agujeros que se requieren. Los de los lados y la parte inferior de la caja van a recibir los ejes de las ruedas impulsoras. Los agujeros de montaje para el motor y el bloque de la caja de engranajes deben estar en línea a través de la caja. Los dos agujeros de la parte inferior van a recibir los dos pies que equilibran al robot sobre sus ruedas. El agujero de la tapa es para el conector tipo D en el cual se enchufará el cable conector del ordenador. Para cortar los agujeros grandes para la caja de engranajes y los ejes, quite la mayor parte del plástico con un cuchillo caliente o un soldador. Luego, con una pequeña lima, vaya formando el agujero pulcramente hasta alcanzar el tamaño adecuado

Tapa de la caja



TODAS LAS CIFRAS ESTAN EN MILIMETR



CABLES FLEXIBLES DE LOS PARCHES

CONECTOR ENCHUFE D

SENSORES DE CONTACTO
DE MICROINTERRUPTOR

PIES DE EQUILIBRIO

Kevin Jones

Lista de componentes

N.º	Artículo
1	40109
3	Conector DIL de 16 patillas
2	Resistencia de 100 ohmios
2	Resistencia de 0,5 watios 270 ohmios
2	Condensador de 0,1 μF
1	Condensador 25 V 1000 μF
1	Veroboard de 24 franjas × 50 agujeros
1	Bobina de cable estañado
1	Enchufe D 15 vías
1	Conector D 15 vías
1	Cubierta D 15 vías
1	Conector de potencia 2,1 mm
1	Conector IDC 20 vías
1	Conector marginal 24 vías
1	Caja 180 × 110 × 55 mm
1	Taco de tiras autoadhesivas
2	Pie de gabinete
1	Paquete tuercas 2BA
1	Paquete tornillos 6BA de 0,5 pulgadas
1	Paquete tuercas 6BA
1	Paquete tornillos M5 de 25 mm
1	Paquete tuercas M5

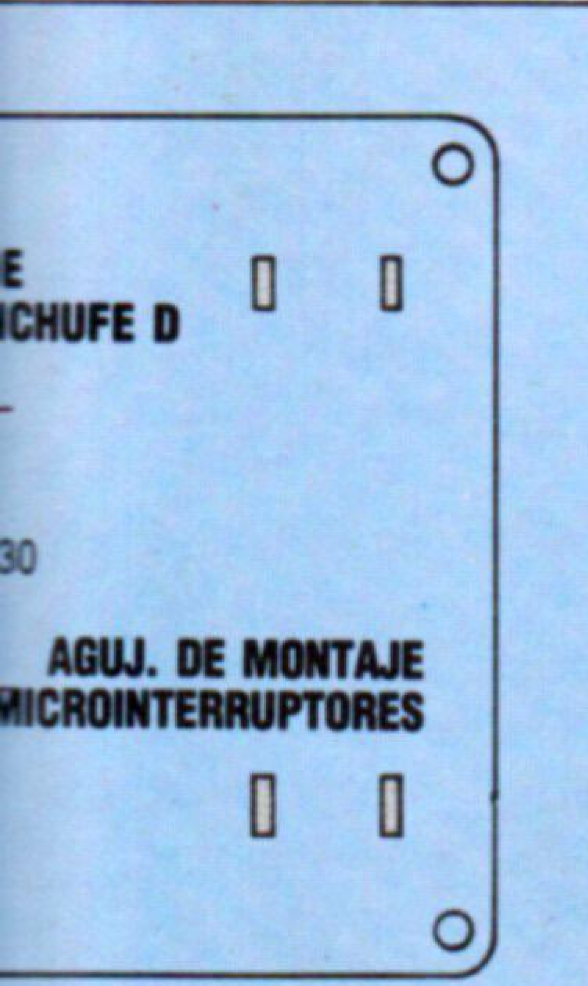
PIEZAS DE RADIO

2	Activador motor paso a paso SAA 1027
2	Motor paso a paso
2	Caja de engranajes sincrónica 25:2

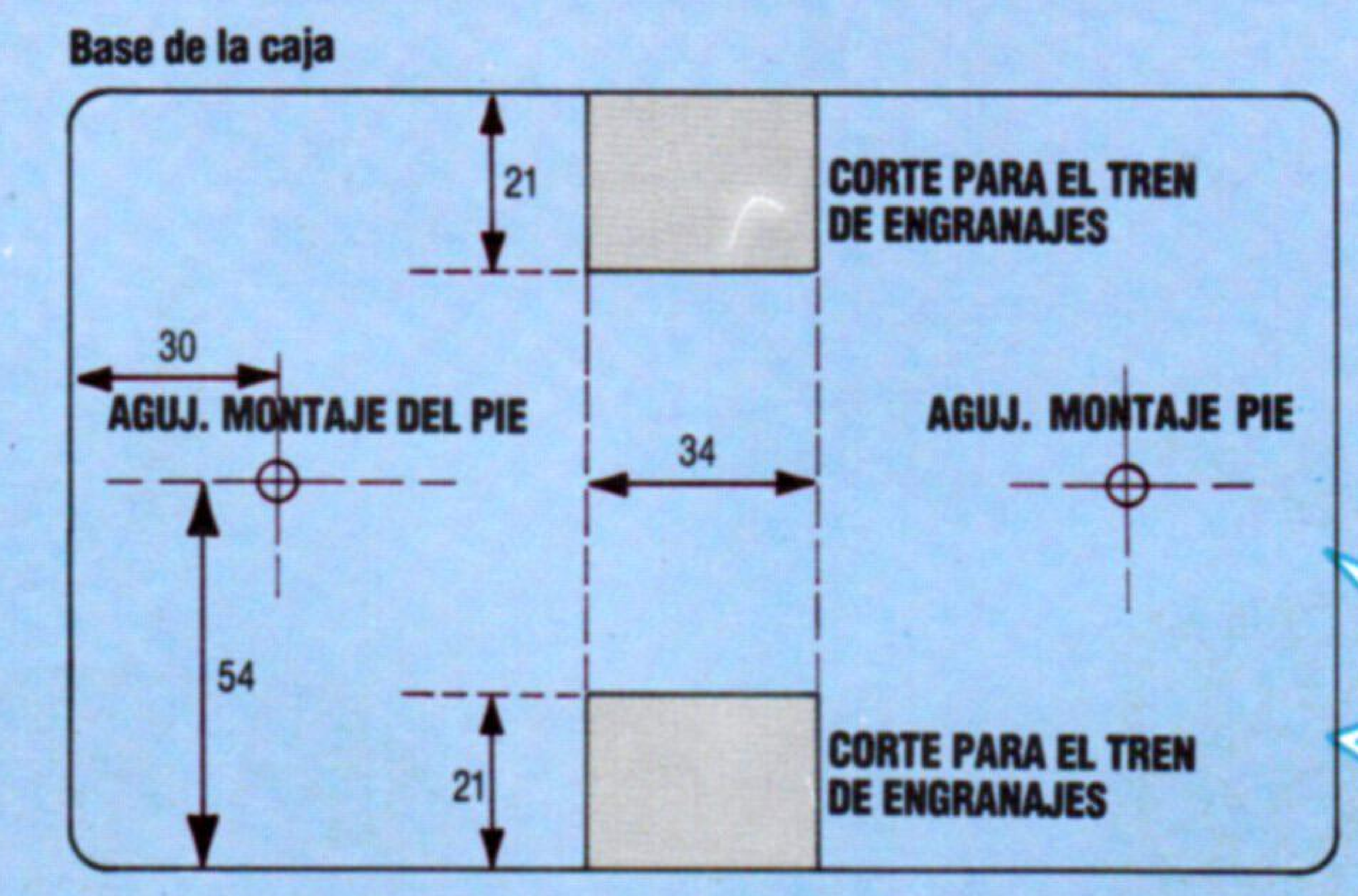
VARIOS

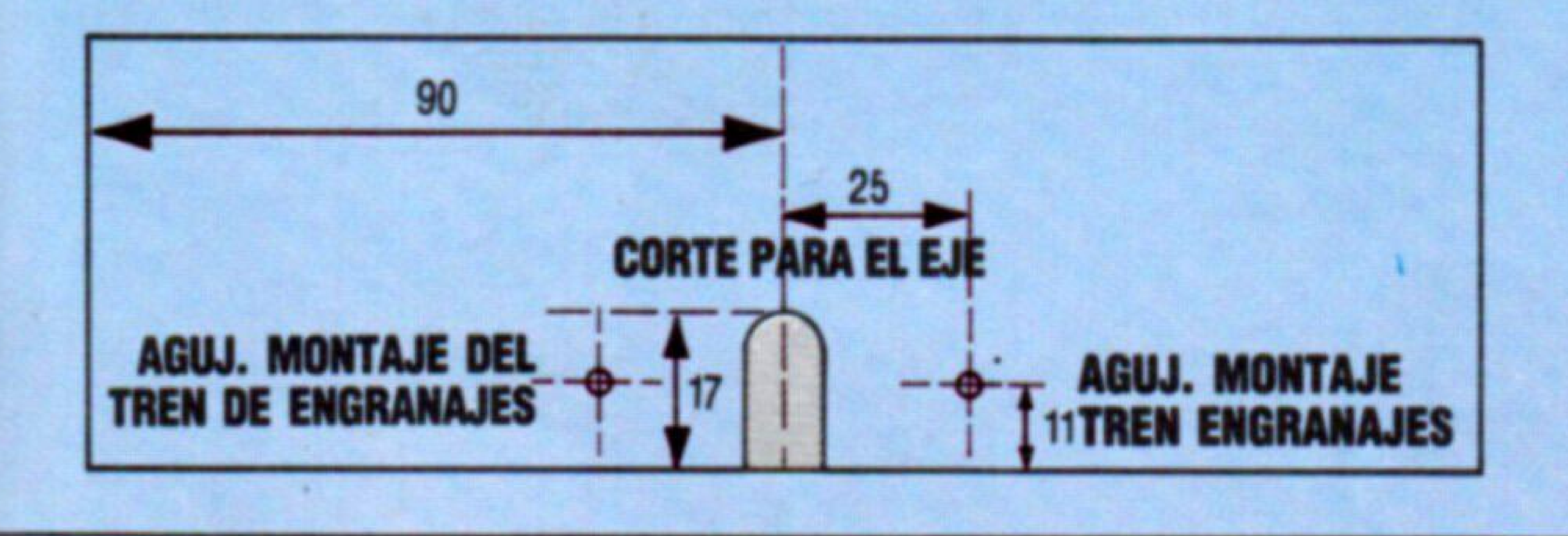
2	Ruedas Lego de 62 mm
1	Paquete ejes technics
4 m	Cable plano de 12 vías
1 m	Cable plano de 20 vías
1	Fuente CD 12 V 1 amp
2	Tornillo 2BA × 4 cm

El importe total de estos componentes puede que sea un poco alto; por este motivo el robot será más atractivo como proyecto de grupo o escolar que como trabajo individual



Base de la caja



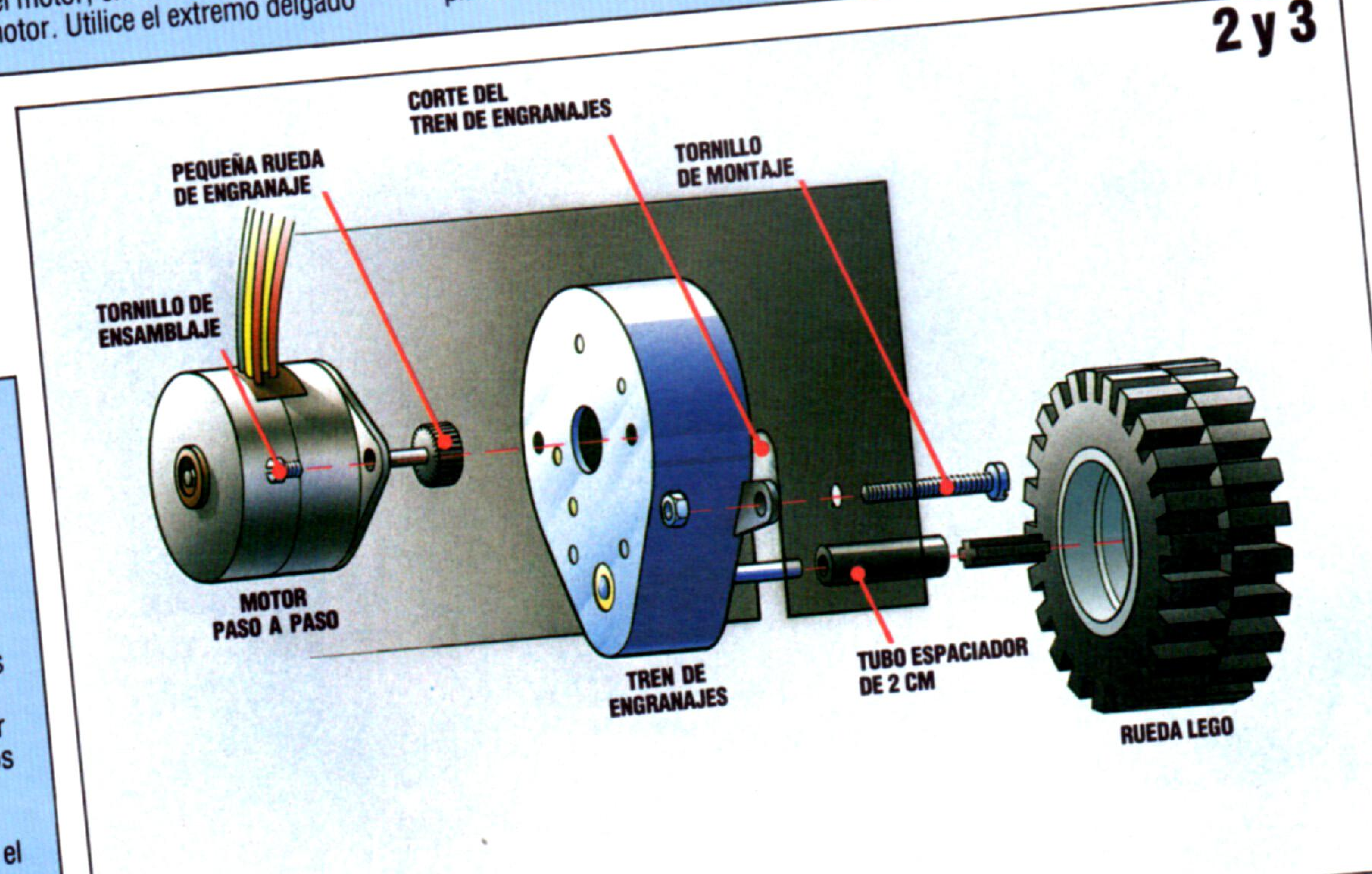


¡ATENCIÓN!

El robot consume una gran cantidad de energía; si la fuente de alimentación ha de activar asimismo la caja buffer, entonces el robot no tendrá potencia suficiente y no se moverá. Por consiguiente, el robot se debe conectar directamente a la puerta para el usuario de su ordenador. Siga fielmente las instrucciones: cualquier error podría dañar su máquina

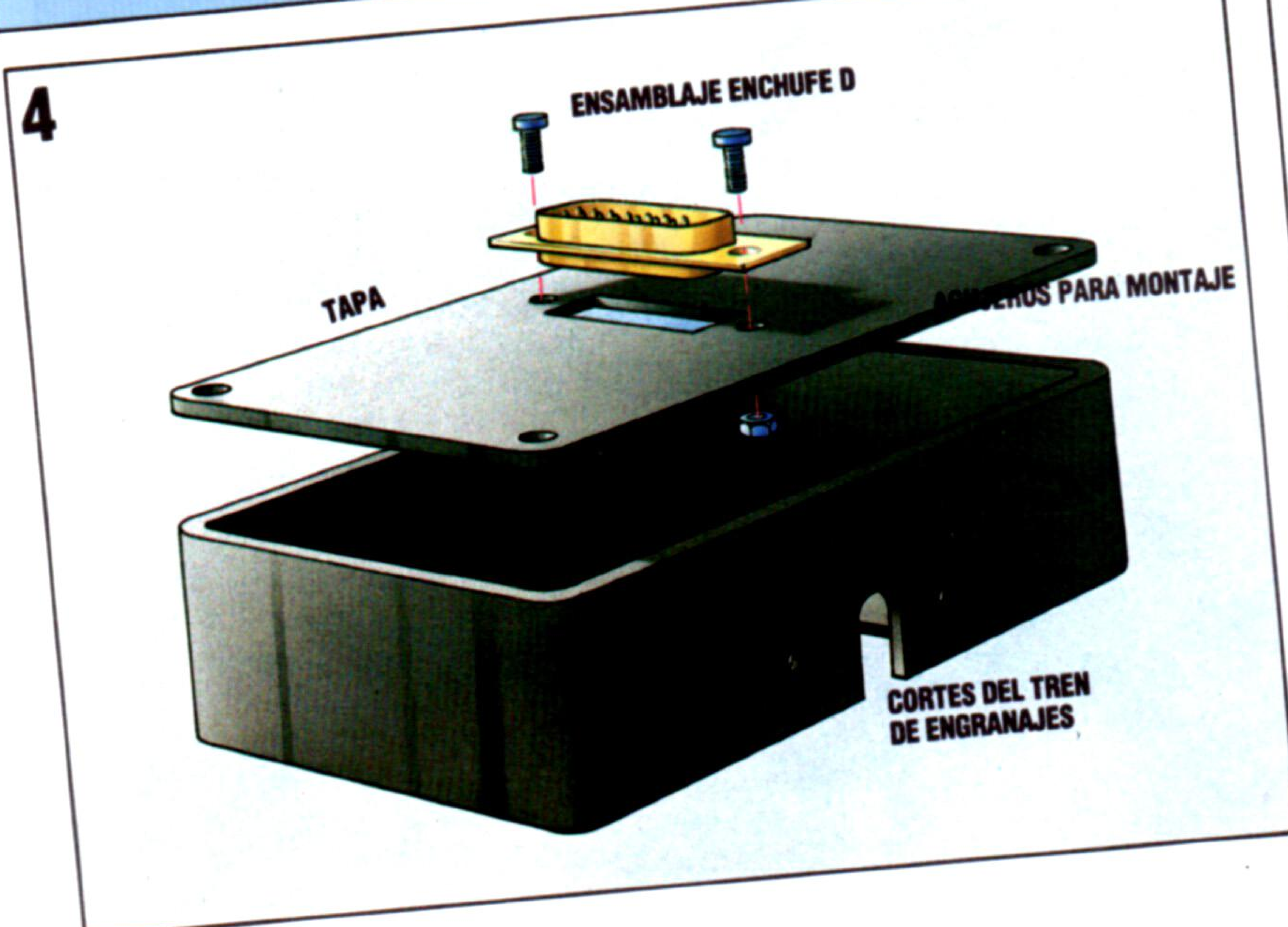
Pasos 2 y 3

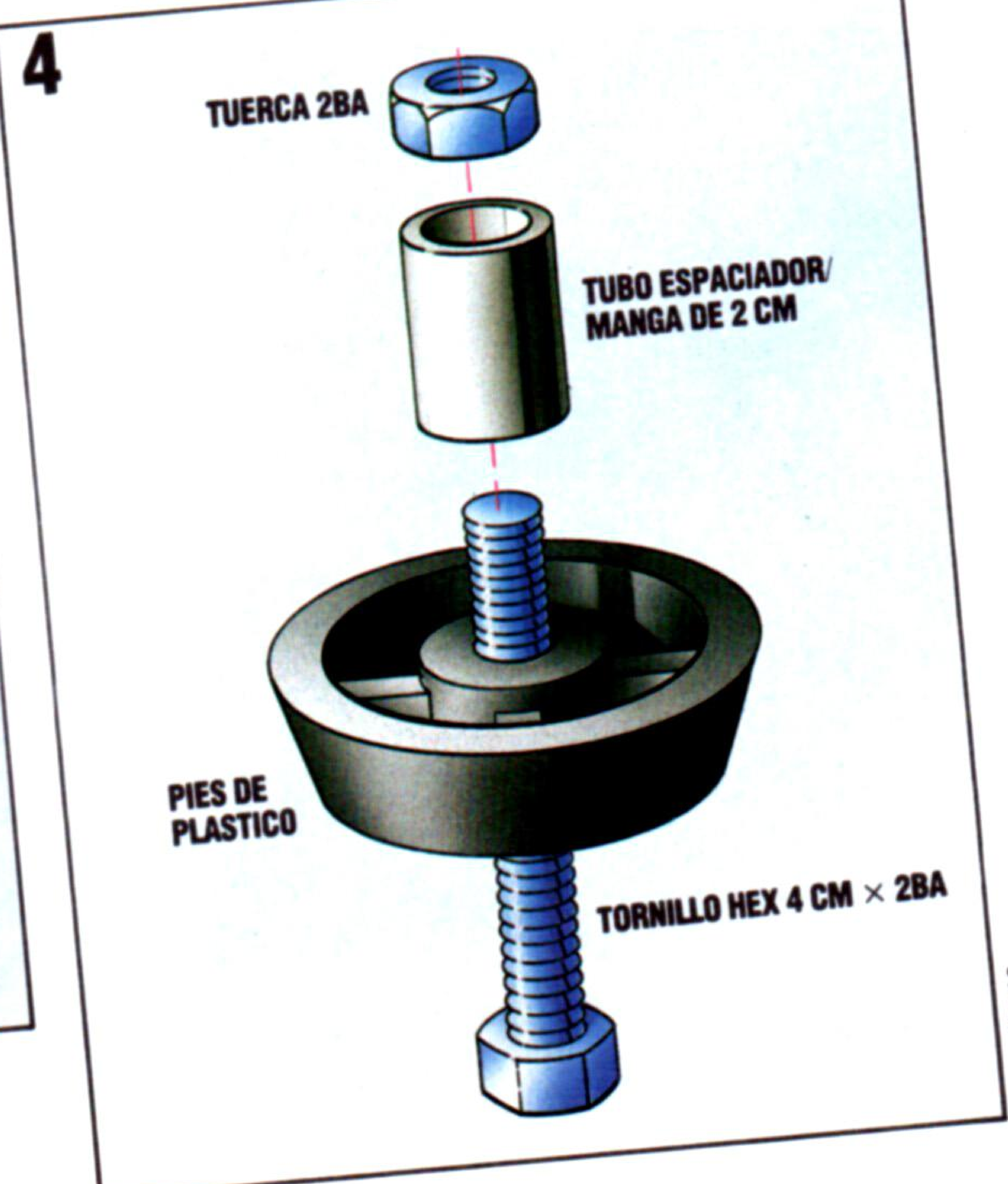
Los motores y la caja de engranajes se venden por separado y se deben ensamblar. La caja de engranajes viene acompañada por un pequeño engranaje de metal suelto y un pequeño tubo espaciador de plástico. El engranaje se debe encajar con la rueda dentada que sobresale del motor. Aplique un poco de pegamento Cyanoacrylate ("supercola") en el diámetro interior a través del engranaje y colóquelo, con el avellanador, en el eje del motor, en el engranaje, alejado del motor. Utilice el extremo delgado del tubo espaciador para alejar el engranaje correctamente del cuerpo del motor, tal como ilustra el diagrama. Deje transcurrir dos o tres horas para que la cola se seque.
Monte el motor (y el engranaje) en la caja de engranajes con los cables del motor hacia el extremo más ancho de la caja. Con este objeto ésta trae dos tornillos. Tenga cuidado de encajar bien el engranaje con los engranajes internos de la caja cuando presione el motor en su sitio. Utilice los tornillos M5 para montar la caja en la carcasa plástica. La caja no se atornilla directamente en la carcasa, sino que se sostiene con el tornillo que se sujeta a ésta. El mismo hace que se puedan ajustar la caja y la rueda.
Las ruedas sólo van montadas en los ejes Lego especiales seccionados en x. Deslice unos 2 cm de la cubierta de plástico de un bolígrafo de diámetro adecuado sobre el husillo de la caja de engranajes como si fuera una manga. Asegúrela con supercola. Ahora pegue un eje Lego en el extremo de la manga. Utilice el más corto de estos ejes. Las ruedas se encajan a "presión" en los correspondientes ejes



Cuarto paso

Atornille el conector D (enchufe) macho en la tapa, con las patillas hacia arriba, utilizando los tornillos y tuercas 6BA. Monte los dos pies de equilibrio, empleando un tornillo 2BA a través de cada pie y ajustándolos en el interior de la carcasa con una tuerca. Los pies deben estar separados 3 cm de la base de los pies, medidos desde la parte inferior de la carcasa. Para ello coloque mangas de 2 cm entre el pie y la parte inferior de la carcasa. O bien se pueden utilizar dos tuercas de fijación





Kevin Jones

Puesta en escena

Ahora diseñaremos rutinas para describir los escenarios del juego y permitir que el jugador se desplace a través de ellos

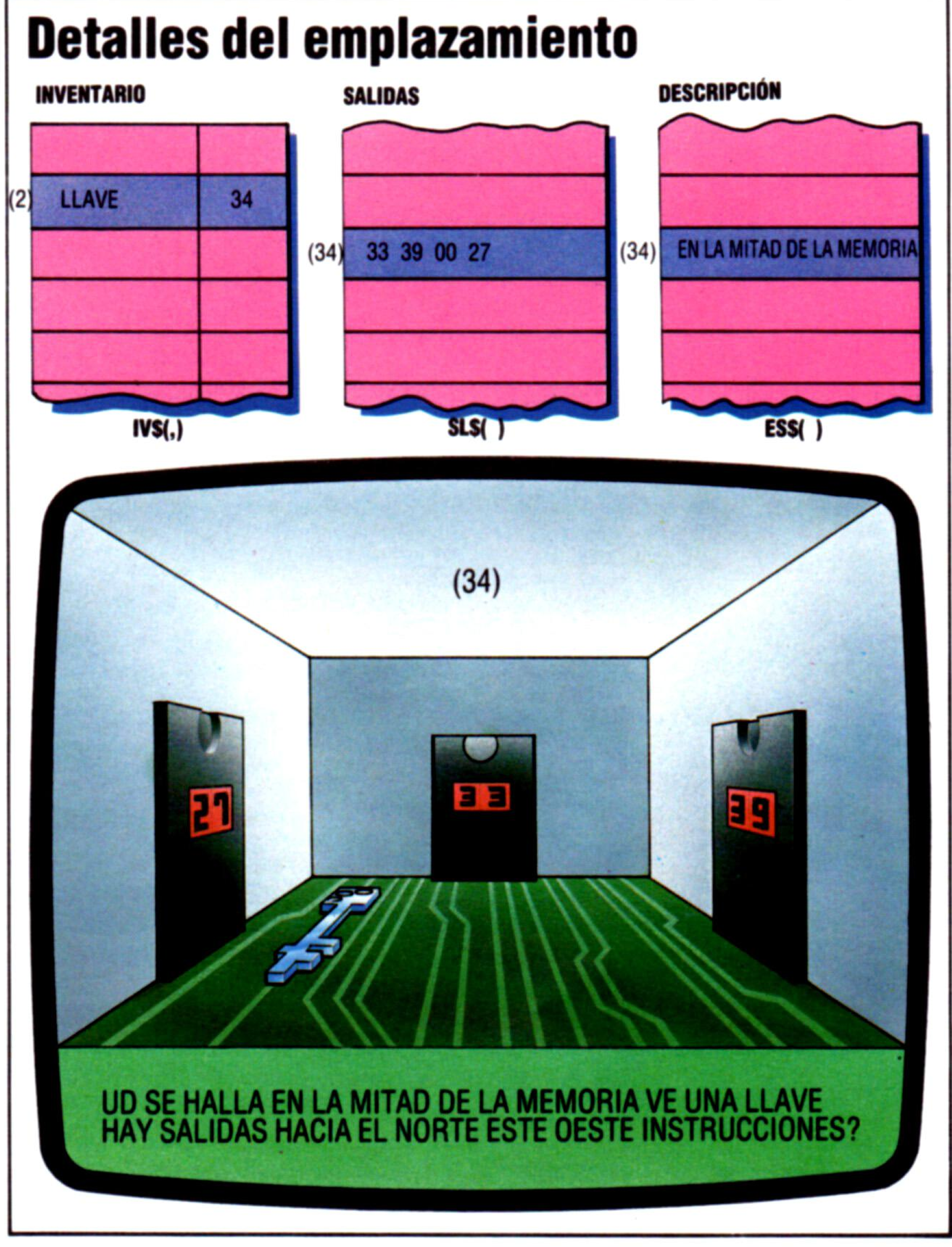
Las descripciones básicas de todos los escenarios están retenidas en la matriz ES$ (véase p. 1247) y a ellas se puede acceder simplemente especificando el número del escenario al que se ha llegado. En *El bosque encantado*, la posición que ocupa el jugador en cualquier momento dado se almacena en la variable P y, por consiguiente, la descripción de ese emplazamiento está almacenada en ES$(P). Cuando se diseñaron originalmente los datos de los escenarios, se tuvo presente el contexto gramatical final de la descripción; ésta se redactó siempre de modo tal que pudiera ir precedida por "Ud. se halla...". Para un escenario dado, P, la descripción se puede formatear y visualizar mediante la utilidad desarrollada en el capítulo anterior, combinando "Ud. se halla" con la descripción retenida para ese emplazamiento en la matriz ES$(). Esto se puede apreciar en la línea 2010 del listado de *El bosque encantado*.

Además de la descripción básica del lugar al cual ha llegado, el jugador deseará saber también si ahí hay algún objeto. Los objetos utilizados en el juego están almacenados (junto con sus posiciones iniciales en el inventario) en una matriz bidimensional, IV$(,). Por ejemplo, IV$(N,1) contiene la descripción del objeto N del inventario, y IV$(N,2) contiene su posición. Si queremos determinar si en un emplazamiento determinado hay o no un objeto, debemos buscar en el inventario, cotejando la posición de cada objeto con el número del escenario que se está describiendo. Dado que en *El bosque encantado* sólo hay tres objetos, y ocho en *Digitaya*, se puede implementar una simple búsqueda lineal empleando un bucle FOR...NEXT.

El bucle de búsqueda utilizado en *El bosque encantado* está entre las líneas 2040-2080. Se explora la segunda columna de la matriz del inventario en busca de un emparejamiento con la posición en curso P. Cuando se encuentra una coincidencia, entonces se agrega la descripción correspondiente a la frase que describe los objetos. Puesto que en cualquier escenario en un momento dado podría haber más de un objeto, debemos dar lugar a la construcción de una frase en la cual se ofrezca una lista de objetos, separados cada uno por una coma. Mediante el empleo de SP$, inicialmente como una serie nula y luego como una coma, podemos insertar la puntuación correcta entre cada ítem. Un flag, F, establecido inicialmente en cero, se pone a uno para indicar el hecho de que en el transcurso de la búsqueda se ha encontrado un emparejamiento. Si al final de la búsqueda el flag continúa a cero, ello significa que no hay ningún objeto presente, y este hecho se le puede transmitir al jugador, como ocurre en la línea 2090 de *El bosque encantado*.

```
2000 REM **** DESCRIBIR ESCENARIO ****
2010 SN$="Ud. SE HALLA "+ES$(P):GOSUB5500
2020 SN$="VE"
2030 REM ** VERIFICAR INVENTARIO PARA OBJ **
2040 F=0:SP$=" "
2050 FOR I=1 TO 3
2060 IF VAL(IV$(I,2))<>P THEN 2080
2070 SN$=SN$+SP$+"UN "+IV$(I,1):F=1:SP$=", "
2080 NEXT I
2090 IF F=0 THEN SN$="NO "+SN$+"NINGUN OBJETO"
2100 GOSUB5500:REM FORMATEAR SALIDA
2110 RETURN
```

Los datos que contienen los detalles relativos a las posibles salidas desde cada escenario están retenidos en la matriz SL$(). Cada valor de la serie está compuesto por ocho dígitos. Subdividiendo estos ocho dígitos en grupos de dos, obtenemos (de izquierda a derecha) los números de los escenarios que hay hacia el norte, el este, el sur y el oeste del escenario en curso. Con el fin de determinar las posibles salidas, el programa primero divide la serie



Habitación con llave
Los detalles de los escenarios están almacenados en tres matrices en serie, que contienen nombres de objetos y su ubicación (V$), las salidas desde los escenarios (SL$) y descripciones (ES$). SL$(34), por ejemplo, podría contener el número de ocho dígitos 33390027, mostrando que el escenario 34 está conectado con los escenarios 33, 39 y 27. ES$(34) contiene "En la mitad de la memoria", que describe al escenario 34. IV$(2,2) contiene el número 34, indicando que IV$(2,1) (La llave) está en el escenario 34.

Kevin Jones

de ocho dígitos en los cuatro números que describen qué escenario hay en cada dirección.

```
2300 REM **** S/R DESCRIBIR SALIDAS ****
2310 SL$=SL$(P)
2320 NR=VAL(LEFT$(SL$,2))
2330 ES=VAL(MID$(SL$,3,2))
2340 SU=VAL(MID$(SL$,5,2))
2350 OE=VAL(RIGHT$(SL$,2))
```

Si en una dirección dada no hay salida, el valor asignado es cero: y esto es de gran ayuda para la descripción de las salidas. Se debe efectuar una verificación preliminar para ver si hay alguna salida posible antes de empezar a construir la oración "Hay salidas hacia el...". Esto se puede hacer llevando a cabo un OR lógico sobre las cuatro variables de dirección, y esto sólo producirá un resultado de cero si las cuatro variables de dirección fueran cero. Si no resultara de esta forma, entonces la rutina continúa para comprobar cada una de las variables de dirección de una en una. Si la variable no es cero, se le añade a la frase la dirección correspondiente.

```
2355 IF(NR OR ES OR SU OR OE)=0 THEN RETURN
2360 PRINT:SN$="HAY SALIDAS POR EL "
2370 IF NR<>0 THEN SN$=SN$+"NORTE "
2380 IF ES<>0 THEN SN$=SN$+"ESTE "
2390 IF SU<>0 THEN SN$=SN$+"SUR "
2400 IF OE<>0 THEN SN$=SN$+"OESTE "
2410 GOSUB 5500:REM FORMATEAR
2415 PRINT
2420 RETURN
```

Desplazamientos

Ahora que ya hemos desarrollado rutinas que describen cada escenario, podemos desarrollar procedimientos que le permitan al jugador hacer cosas en el mundo que hemos creado. En un futuro capítulo del proyecto estudiaremos algoritmos más detallados que analizan instrucciones. De momento nos ocuparemos de las instrucciones de movimiento que puede impartir el jugador simplemente entrando una instrucción de una sola palabra, como "NORTE" o "SUR". Si una de tales instrucciones se le pasa a una subrutina de movimiento como la variable NN$, entonces la rutina de movimiento es la siguiente:

```
3500 REM **** S/R MOVIMIENTO ****
3510 MF=1:REM ESTABLECER FLAG MOVIMIENTO
3520 DR$=LEFT$(NN$,1)
3530 IF DR$<>"N"ANDDR$<>"E"ANDDR$<>"S"ANDDR$
     <>"O"THEN GOTO3590
3540 IF DR$="N"AND NR<>0 THEN P=NR:RETURN
3550 IF DR$="E"AND ES<>0 THEN P=ES:RETURN
3560 IF DR$="S"AND SU<>0 THEN P=SU:RETURN
3570 IF DR$="O"AND OE<>0 THEN P=OE:RETURN
3580 PRINT:PRINT"NO PUEDES ";IS$
3585 MF=0:RETURN
3590 REM ** NOMBRE NO DIRECCION **
3600 PRINT"QUE ES ";NN$;" ?"
3610 MF=0:RETURN
```

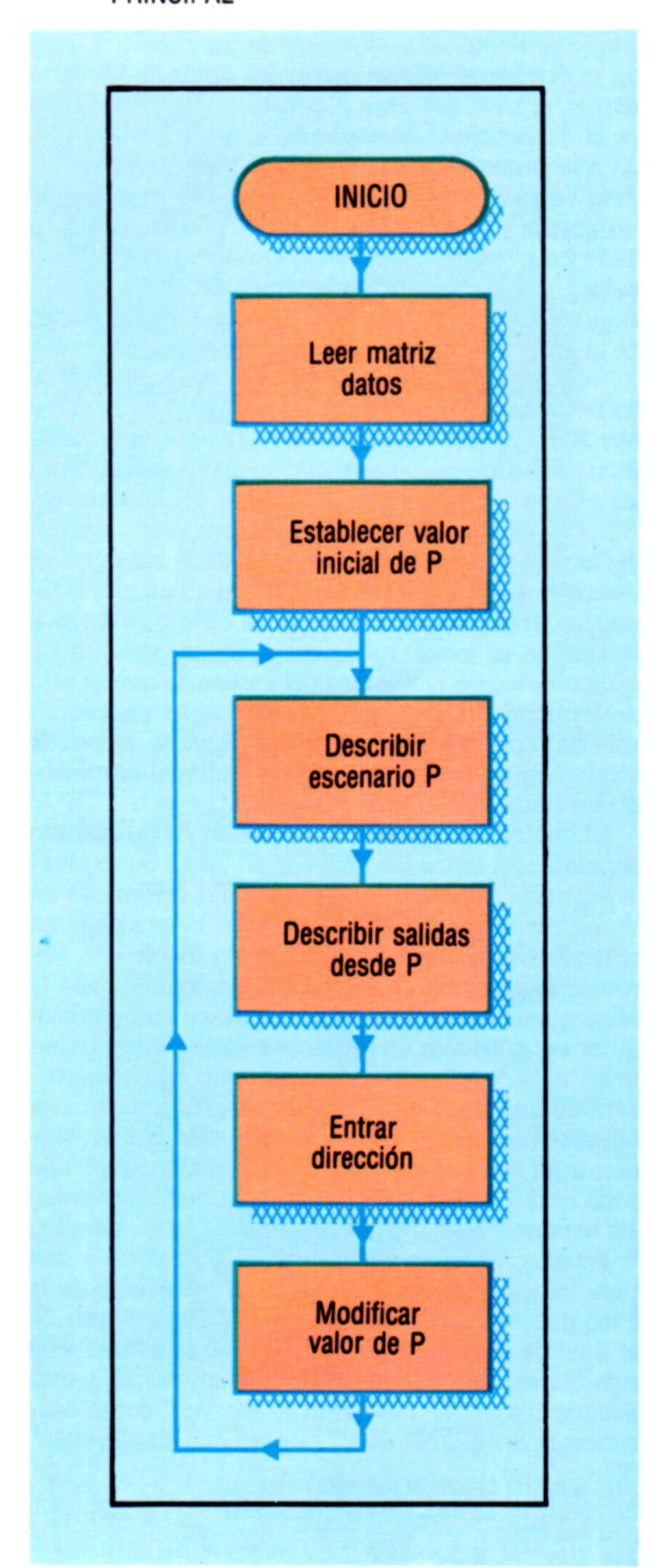
Esta rutina en realidad sólo emplea la primera letra de la instrucción de dirección que se le pasa. Comienza por comprobar que la instrucción sea realmente una dirección. De ser así, se actúa sobre la dirección especificada en la instrucción. Después de asegurarse de que hay una salida en esa dirección, se cambia P (la variable que lleva el registro de la posición del jugador) por el valor de NR, ES, SU u OE.

Sin embargo, antes de que podamos utilizar las subrutinas que hemos desarrollado aquí, necesitamos unirlas para que conformen un bucle repetitivo. El diagrama de flujo ilustra la estructura lógica de este bucle principal de llamada. Si bien ésta no es la estructura final del bucle principal del programa, sirve para demostrar los aspectos que hemos cubierto hasta ahora. Para utilizar las subrutinas dadas aquí, inserte las siguientes líneas, que constituyen una parte del bucle principal.

```
200 GOSUB6000:REM LEER DATOS MATRIZ
210 P=INT(RND(TI)*10+1):REM PUNTO DE PARTIDA
230 REM **** AQUI EMPIEZA EL BUCLE PRINCIPAL ****
240 MF=0:REM FLAG MOVIMIENTO
245 PRINT
250 GOSUB2000:REM DESCRIBIR POSICION
255 GOSUB2300:REM DESCRIBIR SALIDAS
260 PRINT:INPUT"INSTRUCCIONES";IS$
```

También incluya las siguientes líneas en el bucle de llamada principal:

```
270 NN$=IS$:GOSUB 3500:REM MOVIMIENTO
280 GOTO 230:REM RECOMENZAR BUCLE
    PRINCIPAL
```



Para el Spectrum

Debido a que el Spectrum retiene todas las matrices en serie como series de longitud fija, surgen problemas cuando deseamos imprimir uno de los elementos de una matriz en serie como parte de una frase más larga. Cuando se dimensiona una matriz en el Spectrum, el último número de la sentencia define la longitud de cada elemento de la matriz. Por ejemplo, DIM a$(3,2,20) dimensiona una matriz de tres por dos elementos, teniendo cada elemento una longitud fija de 20 caracteres. Si asignamos un elemento de la matriz a una serie de menos de 20 caracteres, entonces la diferencia se compensa agregando espacios al final de la serie. Esto supone un precioso espacio de memoria. Por lo tanto, debemos antes que nada eliminar cualquier espacio en cola. Para hacer esto, los usuarios del Spectrum deben digitar la siguiente rutina en el listado de *El bosque encantado*:

```
7000 REM **** PODADO SPECTRUM ****
7010 FOR I=LEN(A$)TO 1 STEP −1
7020 IF A$(I TO I)<>" " THEN LET N=I:LET I=1
7030 NEXT I
7040 LET S$=S$+A$(TO N)
7050 RETURN
```

Para el listado del *Digitaya*, digite estas mismas instrucciones pero emplee los números de línea del 8500 al 8550.

Esta rutina trunca A$, eliminando todos los espacios posteriores, antes de agregársela a S$. Recuerde que S$ es la variable utilizada para ensamblar una frase para su formateado. Para emplear esta rutina debemos pasar a la variable A$ el elemento de la matriz en serie (a incorporar a la oración), y después llamar a la subrutina. Por consiguiente, en las versiones de *El bosque encantado* y *Digitaya* para el Spectrum, hemos de introducir estos cambios:

El bosque encantado:

```
2010 LET S$="UD SE HALLA":A$=L$(P):
     GOSUB7000:GOSUB5500
2070 LET S$=S$+P$+"UN ":A$=V$(I,1):
     GOSUB7000:LET F=1:LET P$=", "
```

Digitaya:

```
1450 LET S$="UD SE HALLA":A$=L$(P):
     GOSUB8500:GOSUB5880
1500 IF VAL(V$(I,2))=P THEN LET
     S$=S$+P$+"UN ":A$=V$(I,1):
     GOSUB8500:LET F=1:LET P$=" "
```

Listado "Digitaya"

La estructura de *Digitaya* es similar a la de *El bosque encantado*. Agrégueles las siguientes líneas a los listados que hemos ofrecido hasta ahora:

```
1100 GOSUB 6090:REM LEER DATOS MATRIZ
1210 PRINT:INPUT"INSTRUCCIONES";IS$
1120 P=47:REM PUNTO DE PARTIDA
1130 :
1140 REM **** AQUI EMPIEZA EL BUCLE PRINCIPAL ****
1150 :
1160 MF=0:PRINT
1170 GOSUB1440:REM DESCRIBIR ESCENARIO
1180 GOSUB1560:REM LISTAR SALIDAS
```

Incluya, asimismo, estas líneas:

```
1220 NN$=IS$:GOSUB 2000:REM MOVER
1230 GOTO 1140:REM REINICIAR BUCLE
     PRINCIPAL
```

Describir escenario y salidas

```
1440 REM **** S/R DESCRIBIR POSICION ****
1450 SN$="UD SE HALLA "+ES$(P):GOSUB5880
1460 SN$="VE "
1470 REM ** BUSQUEDA DE OBJETO **
1480 F=0:SP$=" "
1490 FOR I=1 TO 8
1500 IF VAL(IV$(I,2))=P THEN SN$=SN$+SP$+"UN"
     +IV$(I,1):F=1:SP$="  "
1510 NEXT I
1520 IF F=0 THEN SN$="NO "+SN$+"NINGUN OBJETO"
1530 GOSUB5880:REM FORMATEAR
1540 RETURN
1550 :
1560 REM **** S/R LISTA SALIDAS ****
1570 SL$=SL$(P)
1580 NR=VAL(LEFT$(SL$,2))
1590 ES=VAL(MID$(SL$,3,2))
1600 SU=VAL(MID$(SL$,5,2))
1610 OE=VAL(RIGHT$(SL$,2))
1620 IF(NR OR ES OR SU OR OE)=0THEN RETURN
1630 PRINT:SN$="HAY SALIDAS HACIA EL "
1640 IF NR<>0 THEN SN$=SN$+"NORTE "
1650 IF ES<>0 THEN SN$=SN$+"ESTE "
1660 IF SU<>0 THEN SN$=SN$+"SUR "
1670 IF OE<>0 THEN SN$=SN$+"OESTE "
1675 GOSUB 5880:REM FORMATEAR
1680 PRINT:RETURN
```

Subrutina Moverse a

```
2000 REM **** S/R MOVERSE A ****
2010 MF=1:REM ESTABLECER FLAG MOVIMIENTO
2020 DR$=LEFT$(NN$,1)
2030 IFDR$<>"N"ANDDR$<>"E" ANDDR$<>"S"ANDDR$<>"O"
     THEN2100
2040 IF DR$="N" AND NR<>0 THEN P=NR:RETURN
2050 IF DR$="S" AND SU<>0 THEN P=SU:RETURN
2060 IF DR$="E" AND ES<>0 THEN P=ES:RETURN
2070 IF DR$="O" AND OE<>0 THEN P=OE:RETURN
2080 PRINT"NO PUEDES ";IS$
2090 MF=0:RETURN
2100 REM NOMBRE INCORRECTO
2110 PRINT"QUE ES ";NN$;" ?"
2120 MF=0:RETURN
```

Complementos al BASIC

Spectrum:
En los listados para ambos juegos, reemplace SL$() por E$(), SL$ por X$, SN$ por S$, IS$ por T$, ES$() por L$(), NN$ por R$, SP$ por P$, DR$ por D$. En el listado de *Digitaya* sustituya las siguientes líneas:

```
1580 LET NR=VAL(X$(TO 2))
1590 LET ES=VAL(X$(3 TO 4))
1600 LET SU=VAL(X$(5 TO 6))
1610 LET OE=VAL(X$(7 TO))
2020 LET D$=R$(TO 1)
```

En el listado de *El bosque encantado*, sustituya las siguientes líneas:

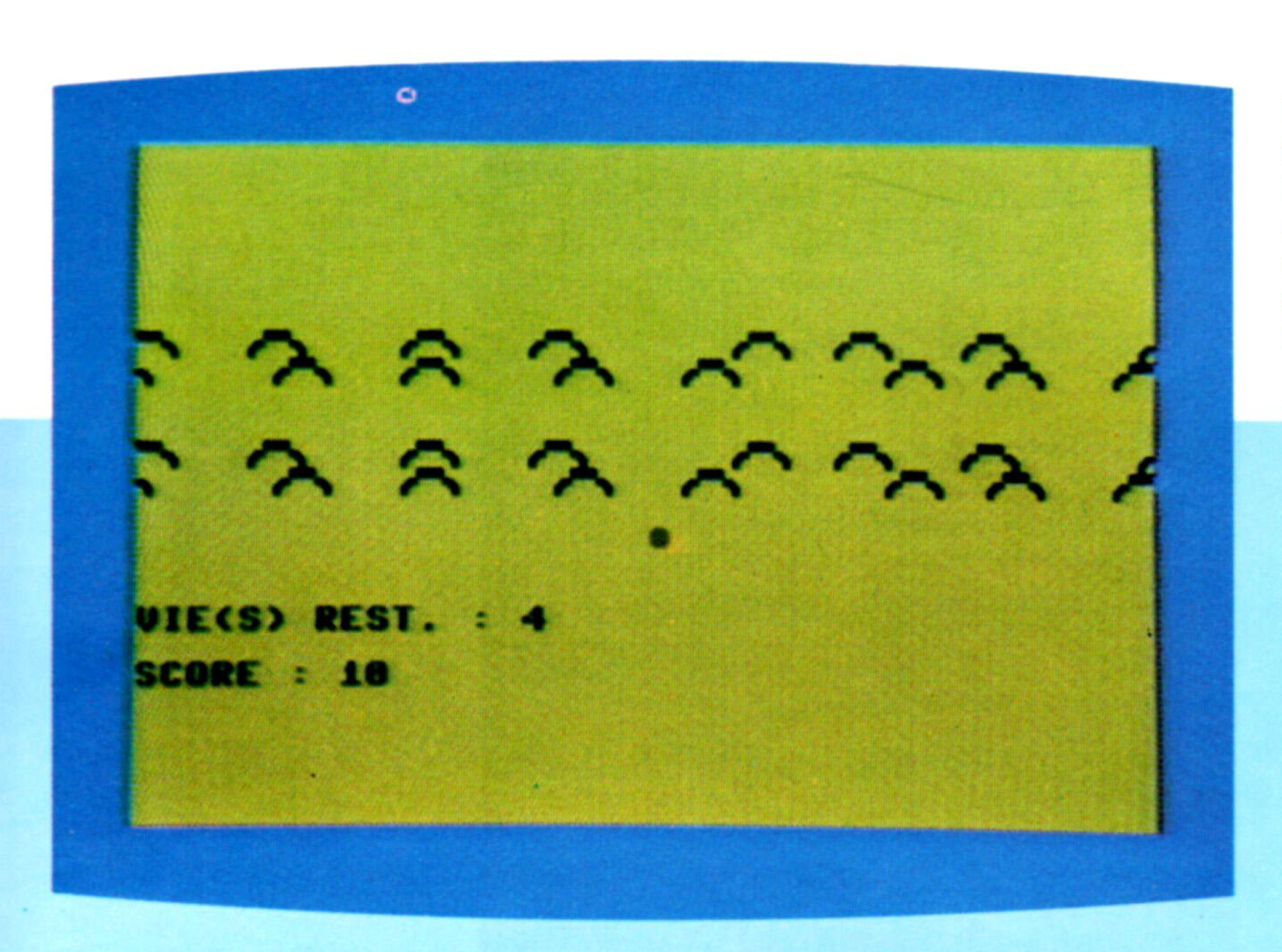
```
2310 RANDOMISE:P=INT(RND(1)*10+1)
2320 LET NR=VAL(X$(TO 2))
2330 LET ES=VAL(X$(3 TO 4))
2340 LET SU=VAL(X$(5 TO 6))
2350 LET OE=VAL(X$(7 TO))
3520 LET D$=R$(TO 1)
```

BBC Micro:
En el listado de *El bosque encantado*, realice la siguiente sustitución de línea:

```
2310 P=RND(10)
```

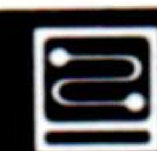
Cangrejos

He aquí la versión para el Commodore 64 de este entretenido juego



Usted debe ayudar a un pobre camarón a regresar al mar, evitando a los voraces cangrejos que pueblan la playa. Cada camarón que logre adentrarse en el agua le proporcionará 10 puntos. Dispone de cinco rutas para intentar obtener su puntuación máxima. Utilice las teclas W para avanzar y Z para retroceder.

```
5 REM *************
10 REM * CANGREJOS *
15 REM *************
20 GOTO 1000
50 PRINT N$;B$;
60 PRINT N$;A$;
70 PRINT BB$;BB$;
80 PRINT N$;B$;
90 PRINT N$;A$;
100 A$=RIGHT$(A$,1)+LEFT$(A$,39)
110 B$=RIGHT$(B$,39)+LEFT$(B$,1)
120 GET X$
130 P=P+40*((X$="W")-(X$="Z"))
140 IF P>PI THEN P=PI
150 IF P=PA THEN 2000
160 C=PEEK(P)
170 IF C<>32 AND C<>CP THEN 3000
180 POKE P1,CR
185 POKE P+N,9
190 POKE P,CP
200 P1=P
210 FOR I=1 TO 6
220 PRINT HH$;
230 NEXT I
240 T=T+1
250 IF T=500 THEN 3900
260 GOTO 50
1000 POKE 53280,6
1010 POKE 53281,7
1015 PRINT CHR$(147);
1020 X$=" "
1030 A$=" "
1040 B$=" "
1045 N$=CHR$(144)
1046 N=54272
1050 PA=1244
1060 PI=1604
1070 P=PI
1080 P1=P
1090 CP=81
1100 CR=32
1110 HH$=CHR(145)
1120 BB$=CHR$(17)
1150 T$=CHR$(19)
1160 RESTORE
1165 FOR I=1 TO 40
1170 READ A
1180 A$=A$+CHR$(A)
1190 NEXT I
1200 B$=A$
1260 NP=5
1265 T=0
1270 S=0
1280 X=INT(RND(TI)*37)+2
1290 A$=RIGHT$(A$,X)+LEFT$(A$,40-X)
1300 PRINT T$;
1310 FOR I=1 TO 17
1320 PRINT BB$;
1330 NEXT I
1340 PRINT N$;"VIDA(S)[1SPC]REST.[1.SPC]:";NP
1350 PRINT
1360 PRINT N$;"PUNTOS[1SPC]:";S
1380 FOR I=1 TO 13
1390 PRINT HH$;
1400 NEXT I
1410 GOTO 50
2000 S=S+10
2010 POKE P1,CR
2020 P=PI
2030 P1=P
2040 GOTO 1280
3000 POKE P1,CR
3020 P=PI
3030 P1=P
3100 VO=54296
3110 WA=54276
3120 AA=54277
3130 HF=54273
3140 LF=54272
3150 SS=54278
3160 PH=54275
3170 PL=54274
3180 POKE VO,15
3190 POKE WA,65
3200 POKE AA,190
3210 POKE PH,15
3220 POKE PL,15
3300 FOR I=1 TO 11
3310 READ H
3320 READ L
3340 READ DL
3350 GOSUB 3700
3360 NEXT I
3370 RESTORE
3380 FOR I=1 TO 40
3390 READ A
3400 NEXT I
3500 NP=NP-1
3510 IF NP=0 THEN 4000
3520 GET X$
3530 GOTO 1280
3700 POKE HF,H
3710 POKE LF,L
3720 POKE SS,136
3800 FOR J=1 TO DL
3810 NEXT J
3820 POKE HF,0
3830 POKE LF,0
3840 RETURN
3900 PRINT CHR$(147);
3910 FOR I=1 TO 12
3920 PRINT
3925 NEXT I
3930 PRINT TAB(12"**[1SPC]TIEMPO[1SPC] TRANSCURRIDO[1SPC]**"
3940 FOR I=1 TO 1000
3950 NEXT I
4000 IF S>R THEN R=S
4010 NB=0
4020 PRINT CHR$(147);
4030 FOR I=1 TO 7
4040 PRINT
4050 NEXT I
4060 PRINT TAB(16);"PUNTOS[1SPC]:";
4070 PRINT S
4080 FOR I=1 TO 4
4090 PRINT
4100 NEXT I
4110 PRINT TAB(12)"PUNTUACION[1SPC]MAXI MA[1SPC]:";
4120 PRINT R
4130 FOR I=1 TO 4
4140 PRINT
4150 GET X$
4160 NEXT I
4170 PRINT TAB(16);"OTRA[1SPC]?"
4180 GET X$
4190 IF X$=" " THEN 4180
4200 END
10000 DATA 117,184,105,32,32,117,184,105
10010 DATA 32,32,32,117,184,105,32,32
10020 DATA 117,184,105,32,32,32,32,32
10030 DATA 117,184,105,32,117,184,105
10040 DATA 32,32,117,184,105,32,32,32,32
20000 DATA 8,147,400,8,147,300
20010 DATA 8,147,100,8,147,400
20020 DATA 10,60,300,9,159,100
20030 DATA 9,159,300,8,147,100
20040 DATA 8,147,300,8,23,100
20050 DATA 8,147,400
```

Las tres últimas órdenes

Ya sólo nos quedan por diseñar tres órdenes para tener listo nuestro programa depurador, pero antes nos referiremos al mecanismo de interrupción

Antes de ocuparnos de las órdenes debemos tener en cuenta el mecanismo de interrupción, el cual es empleado en el programa original en los puntos de ruptura, donde hemos sustituido una instrucción con el opcode SWI (SoftWare Interrupt). El SWI, al igual que las restantes interrupciones del 6809, es accedido a través de una posición específica de memoria, a saber, la $FFFA. Esto significa que cuando se ejecuta un SWI los registros se guardan en la pila y el procesador carga la dirección de 16 bits que hay en $FFFA y $FFFB en el contador de programas (PC). Desde esa dirección comienza entonces la ejecución. Nuestra misión es cambiar este vector para que señale el punto de entrada de nuestro programa depurador. Aquí surge el problema de que los vectores suelen encontrarse en la ROM. El hecho de que estas direcciones sean fijas significa que el sistema operativo debe tener otros medios para "vectorizar" las interrupciones.

El sistema normal consiste en disponer de una tabla de salto (véase p. 1119) colocada en una zona de trabajo de la RAM, que es una parte de la memoria no disponible generalmente para programas y reservada a ciertos usos del sistema operativo. La dirección señalada en el vector contiene una instrucción JMP (*Jump*: saltar) seguida de una dirección, que por lo general volverá a señalar al sistema operativo. Pero esta dirección puede ser cambiada por la que se requiera, de modo que la primera instrucción ejecutada después de la interrupción de software será una JMP de salto a la dirección de entrada del programa depurador. Hay que cuidar de sustituir el contenido original de la tabla de salto antes de que nuestro programa acabe su ejecución, ya que siempre es posible que el sistema operativo ejecute después un SWI. Vale la pena recordar que el 6809 tiene tres interrupciones de software, y que no hay razón alguna para no usar SWI2 (opcode 10 3F y vector en $FFF4) o SWI3 (opcode 11 3F y vector en $FFF2), aunque el que éstas empleen opcodes de dos bytes hace necesario algunos cambios en el programa depurador.

Otro problema es que nuestro programa sólo puede ocupar cualquier sitio de la memoria que el programa a depurar deje libre. El depurador debe, pues, ser reubicable. De hecho habrá notado que todas las referencias a posiciones de memoria en el programa hecho o por hacer emplean un direccionamiento relativo al contador de programas. Pero en un cierto punto debemos saber la dirección absoluta del punto de entrada del programa para poder colocarlo en la tabla de salto de interrupciones. Tal dirección se tendrá que calcular en tiempo de ejecución, dado que el ensamblador no lo puede hacer.

Nuestra primera tarea será calcular esta dirección e insertarla en la tabla de salto. Se observará que la dirección del punto de entrada para el SWI será distinta de la dirección de inicio del programa depurador, puesto que la rutina existente en la dirección de inicio del programa debe encargarse del procedimiento de inicialización, que no se necesitará cuando se vuelva a entrar en el programa a través del SWI. Según esto, debemos ocuparnos de toda la inicialización dentro de una subrutina; el punto de entrada será la dirección que contiene la instrucción después de la llamada BSR a esta subrutina. Es muy conveniente que esta dirección sea precisamente la que se guardó en la pila por medio de la llamada BSR para que podamos tomarla de la pila y colocarla en el punto adecuado dentro de la tabla de salto.

Otra misión del procedimiento de inicialización es la de obtener la dirección de inicio del programa a depurar.

He aquí su diseño completo:

Procedimiento de inicialización

Data:

Vector-Dirección es la dirección que ha de encontrarse en $FFFA en X
Opcode-JMP es el opcode de la instrucción JMP en A
Dirección-Entrada es la dirección del punto de entrada en Y
Dirección-Inicio del programa a depurar en D

Proceso:

Tomar Vector-Dirección
Almacenar Opcode-JMP en Vector-Dirección
Tomar Dirección-Entrada
Almacenarla en (Vector-Dirección+1)
Tomar Dirección-Inicio del teclado
Guardarla

Ahora ya podemos afrontar la tarea de codificar las tres órdenes restantes. Algo que también debemos tener en cuenta afecta a una de estas órdenes, en concreto la orden R que visualiza los registros. Naturalmente no deseamos visualizar el contenido actual de los registros mientras se ejecuta el programa depurador, sino que queremos inspeccionar su contenido en el momento de ocurrir el punto de ruptura. Quiere decir que lo que nos interesa es saber los valores que la instrucción SWI hizo colocar en la

B	Insertar punto ruptura
U	Desinsertar punto ruptura
D	Visualizar puntos actuales de ruptura
S	Inicio ejecución programa
G	Ir o reanudar el programa desde donde se cortó
R	Visualizar el contenido de los registros
M	Inspeccionar y cambiar la posición de la memoria
Q	Abandonar el programa

pila. Sin embargo, puede haber sucedido que en la pila se hayan colocado encima otros valores cuando vayamos a inspeccionar la pila. Podemos calcular el número de bytes colocados en la pila que no nos interesan y descartarlo para obtener los valores de los registros. Pero es más sencillo guardar el valor del puntero de la pila en el momento de la interrupción y emplearlo después como referencia.

En la codificación de R supondremos que hemos realizado lo anterior y que no hay problema en obtener este contenido de los registros. La estructuración de la rutina es inmediata, sólo tenemos que tomar cada valor, uno tras otro, sin sacarlos de la pila y visualizarlos con etiquetas adecuadas. La única excepción será el valor S, que será el valor anterior a la interrupción y puede obtenerse añadiendo la cantidad precisa al valor guardado de S que empleamos como referencia de los valores de los registros colocados en la pila.

La orden R

Data:

Puntero-Pila es el indicador de la posición superior de la pila después de la interrupción en X
Valor-Byte-Individual retiene los valores de los registros de un solo byte en B
Valor-Byte-Doble retiene los valores de los registros de dos bytes en D
Etiquetas contiene las etiquetas relativas a los 9 regs.

Proceso:

Tomar Puntero-Pila
Cargar CC en Valor-Byte-Individual
Visualizar etiquetas(1), Valor-Byte-Individual
Repetir lo anterior para A, B y DP
Cargar X en Valor-Byte-Doble
Visualizar Etiquetas(5), Valor-Byte-Doble
Repetir lo anterior para Y, U y PC
Sumar 12 al valor original de Puntero-Pila
Visualizar Etiquetas(9), Puntero-Pila

Faltan dos órdenes más: la orden Q, abandonar (*quit*) el programa, que no necesita en sí una rutina especial; y la orden G, reanudar la ejecución tras el punto de ruptura. En este punto debemos sustituir la instrucción SWI que motivó la ruptura con la instrucción original reemplazada por ella y posteriormente devolver el control a dicha instrucción. Es fácil restablecer los registros a su contenido original con el solo empleo de RTI, que los hace salir a todos de la pila. Pero se debe tener en cuidado de que el valor del PC tomado de la pila sea el valor de la instrucción siguiente. Ya que el que deseamos se obtiene añadiendo un uno, debemos ajustar el valor en la misma pila antes de volver.

La orden G

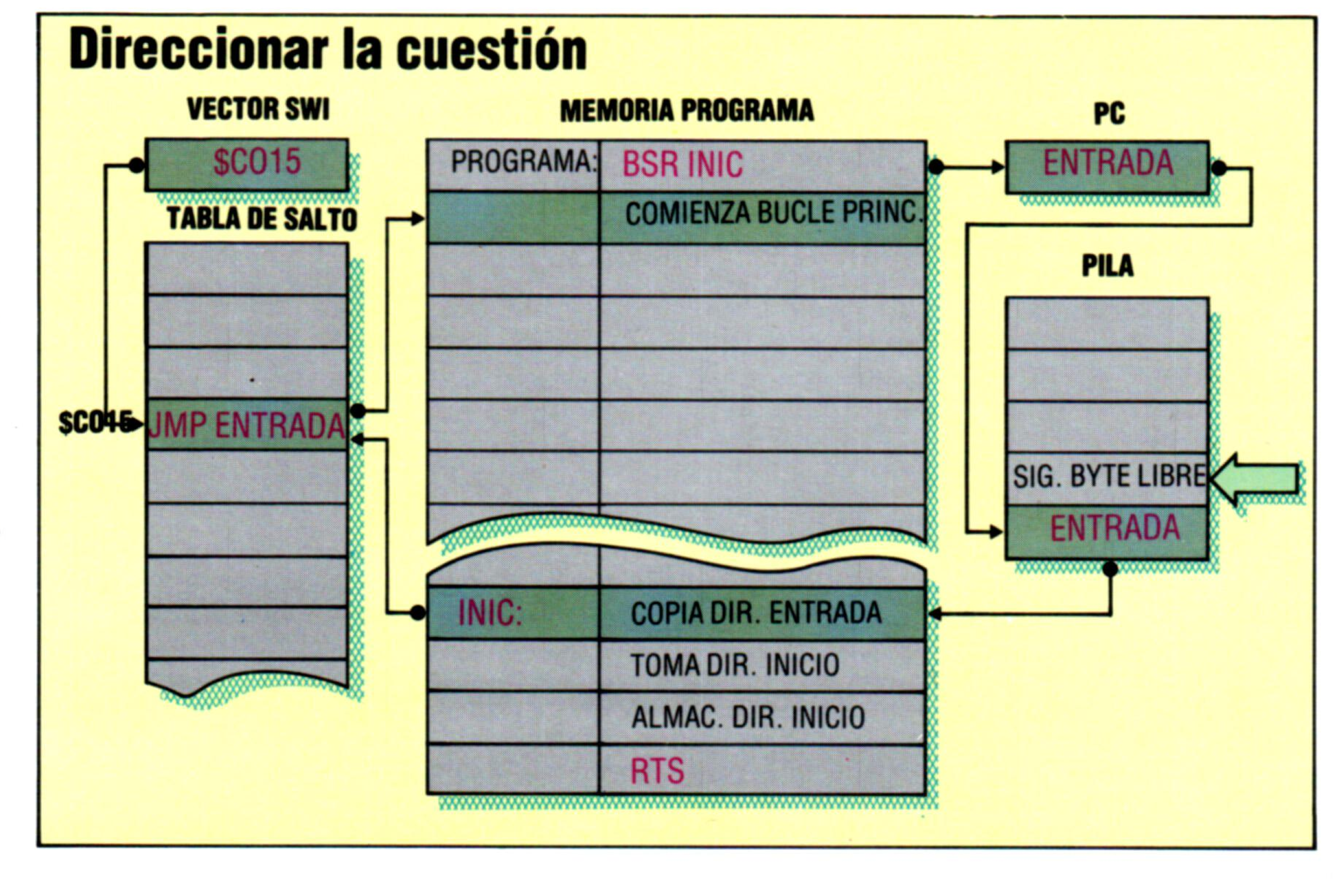
Data:

Tabla-Puntos-Ruptura es tabla de 16 bits de tales pts.
Valores-Sustituidos es una tabla de opcodes sustituidos por instrucciones SWI
Punto-Ruptura-Siguiente es un número entre 1 y 16
Puntero-Pila es el valor guardado del puntero de la pila después de SWI

Proceso:

IF Punto-Ruptura-Siguiente >0 AND <=16 THEN
  Tomar opcode de Valores-Sustituidos (Punto-Ruptura-Siguiente)
  Almacenarlo en la dirección contenida en Tabla-Puntos-Ruptura (Punto-Ruptura-Siguiente)
  Colocar en S el valor de Puntero-Pila
  Decrementar el valor de PC en la pila
  Incrementar Punto-Ruptura-Siguiente
  Retorno de interrupción
ELSE
  Retorno de subrutina

Con la siguiente lección finalizaremos el curso de lenguaje máquina del 6809. En ella codificaremos el módulo principal de nuestro programa depurador y explicaremos cómo opera el programa.



A su debido tiempo
El programa depurador comienza con una llamada BSR a la rutina de inicialización, seguida del comienzo del bucle del programa principal. Una de las tareas de la inicialización es la de determinar la dirección absoluta de este comienzo de bucle y copiarla en la tabla de salto de interrupción, de modo que cuando se ejecute un SWI se pase el control por la tabla de salto y vuelva al comienzo del bucle. Esta dirección no puede conocerse por anticipado dado que el programa ha de ser totalmente reubicable. Por suerte, la dirección de retorno puesta en la pila por BSR es precisamente la dirección en cuestión, lo que quiere decir que la rutina de inicialización sólo necesita copiarla de la pila a la tabla de salto

Procedimiento de inicialización

```
START   RMB   2              Para guardar dir. inicio (start)
OPJMP   FCB   $0E            Opcode-JMP
INIT    LDX   $FFFA          Toma Vector-Dirección
        LDA   OPJMP,PCR      Toma Opcode-JMP y lo guarda
                             en Vector-Dirección
        STA   ,X+
        LDY   1,S            Toma Dir.-Entrada de la pila
        STY   ,X             La guarda en Vector-Dir.+1
        BSR   GETADD         Toma del teclado Dir.-Inicio
        STD   START,PCR      La guarda
        RTS                  Return
```

Orden R

```
STACKP  RMB   2              Puntero-Pila
LABELS  FCC   'CC A BDP X Y
                  UPC S'
SPACE   FCB   32             Espacio en código ASCII
CMDR    PSHS  A,B,X,Y        Guarda registros empleados
        LDX   STACKP,PCR     Toma Puntero-Pila
        LEAY  LABELS,PCR     Emplea Y para apuntar a la
                             etiqueta
        LDA   #4             Núm. de regs. de un solo byte
FOR01   BSR   CMDR1          Visualiza registro siguiente
                             cuatro veces
        DECA
        BGT   FOR01
        LDA   #4             Núm. de regs. de dos bytes
FOR02   BSR   CMDR2          Visualiza registro siguiente
                             cuatro veces
        DECA
        BGT   FOR02
        LDA   ,Y+            Primer carácter de la etiq.
        BSR   OUTCH          Lo visualiza
        LDA   ,Y+            Segundo carácter de la etiq.
        BSR   OUTCH          Lo visualiza
        LDA   SPACE,PCR      Visualiza el espacio
        BSR   OUTCH
        TFR   X,D            X contiene ahora el valor
                             requerido de S
        BSR   DSPADD         Visualiza S
        PULS  A,B,X,Y,PC     Restaura y retorna
* Subrutina para visualizar un registro de un solo byte
CMDR1   PSHS  A              Guarda A
        LDA   ,Y+            Primer carácter de la etiq.
        BSR   OUTCH          Lo visualiza
        LDA   ,Y+            Segundo carácter de la etiq.
        BSR   OUTCH          Lo visualiza
        LDA   SPACE,PCR      Visualiza el espacio
        BSR   OUTCH
        LDB   ,X+            Toma reg. sig. poniéndolo
                             en Valor-Byte-Individual
        BSR   DSPVAL         Visualiza Valor-Byte-Indiv.
        PULS  A,PC           Restaura A y retorna
* Subrutina para visualizar un registro de dos bytes
CMDR2   PSHS  A              Guarda A
        LDA   ,Y+            Primer carácter de la etiq.
        BSR   OUTCH          Lo visualiza
        LDA   ,Y+            Segundo carácter de la etiq.
        BSR   OUTCH          Lo visualiza
        LDA   SPACE,P-CR     Visualiza el espacio
        BSR   OUTCH
        LDD   ,X++           Toma registro siguiente y lo
                             pone en Valor-Doble-Byte
        BSR   DSPADD         Visualiza Valor-Doble-Byte
        PULS  A,PC           Restaura A y retorna
```

Orden G

```
BPTAB   RMB   32             Tabla-Puntos-Ruptura
REMTAB  RMB   16             Valores-Sustituidos
NEXTBP  RMB   1              Punto-Ruptura-Siguiente
CMDG    PSHS  A              Guarda A por si realizamos un
                             retorno normal
        LDA   NEXTBP,PCR     Punto-Ruptura-Siguiente
IF04    BLE   ENDF04         Si Punto-Ruptura-Siguiente>0
        CMPA  MAXBP,PCR      y <=16
        BGT   ENDF04         (núm. máx. de puntos ruptura)
        DECA                 Conv. en desplaz. para tabla
        LEAX  BPTAB,PCR      Dir. de Tabla-Puntos-Rupt.
        LEAY  REMTAB,PCR     Dir. de Valores-Sustit.
        LDB   A,Y            Toma Valor-Sustituido
        LSLA                 Convierte A en desplazamiento
                             para tabla de 16 bits
        STB   [A,X]          Lo almacena en la dirección
                             dentro de Tabla-Puntos-Ruptura
        LDS   STACKP,PCR     Coloca Puntero-Pila en S
        DEC   10,S           Ajusta valor del PC en pila
        INC   NEXTBP,PCR     Incrementa Punto-Rupt.-Sig.
        RTI                  Return desde la interrupción
ENDF04  PULS  A,PC           Restaura y retorna
```

Estrellas en pantalla

"Starfinder", escrito para el BBC Micro, es un paquete educativo diseñado especialmente para astrónomos "amateurs"

Los astrónomos siempre se han encontrado con un problema en particular: el del tiempo atmosférico. Para los astrónomos profesionales es un hecho común pasarse semanas preparándose para un acontecimiento particularmente asombroso, sólo para descubrir que una densa nube oscurece la visión. Éste es el motivo por el cual en la actualidad la mayoría de los observatorios se construyen a gran altitud, en lugares donde exista poca nubosidad, o incluso en el espacio.

Ahora Century Software ha traído el universo hasta la pequeña pantalla a través de *Starfinder*, un paquete que contiene software en cassette y un libro. En esencia, el programa le permite al usuario contemplar cualquier sección del cielo de cualquier lugar del mundo en cualquier momento del s. XX.

Una vez que se ha cargado el programa desde la cassette, se le ofrece al usuario una lista de opciones. Una de ellas es observar el cielo en ese momento; la vista por defecto es la del cielo tal como se veía mirando hacia el sur desde Londres en la medianoche (hora del meridiano de Greenwich) del 21 de noviembre de 1984. Esto parece ser puramente arbitrario, ya que el panorama que ofrecía el cielo en esa fecha no tenía nada de especial: es probable que esto se haya diseñado así para coincidir con la fecha del lanzamiento del paquete y no con ningún acontecimiento cósmico específico.

En la parte superior de la pantalla el programa visualiza la hora y la fecha, y da la situación del observador en términos de longitud y latitud. Debajo de esta información está la carta estelar propiamente dicha, que muestra el cielo nocturno de suroeste a sureste, con una altitud (el ángulo de vista) de 60°. Al programa le lleva algunos segundos visualizar las estrellas, puesto que para procesar y trazar cada uno de los cuerpos celestes se requiere gran cantidad de cálculos numéricos.

Las estrellas se representan como cuadrados blancos (el programa se ejecuta en la modalidad 4) y las estrellas más brillantes se indican con un mayor tamaño. Es una verdadera lástima que las máquinas Acorn no posean una instrucción de "brillo", lo que haría más realista la visualización. Los planetas también tienen forma cuadrada, pero están trazados en rojo, mientras que el Sol se representa mediante un gran cuadrado amarillo y la Luna por medio de un pequeño punto amarillo. Utilizando la "sonda espacial", que se mueve pulsando las teclas para el cursor, se puede identificar cualquiera de las estrellas reflejadas. Cuando se coloca la sonda espacial (una cruz roja) sobre una estrella, aparece arriba del mapa el nombre científico y el nombre corriente de esa estrella, así como sus coordenadas, dadas como una cifra de altitud y de acimut (la altitud se expresa como una cifra positiva o negativa, con el horizonte como cero; el acimut indica los grados este u oeste proa al norte). El panorama se puede cambiar desde el teclado modificando cualquiera de estas cifras.

Regresando al menú principal, el usuario puede modificar la vista para que corresponda a la que se observó en cualquier momento en cualquier lugar del mundo. El programa también se puede utilizar para encontrar un cuerpo celeste determinado: una estrella, un planeta o el cometa Halley.

A pesar de que este paquete es muy amplio, tiene sus limitaciones. La principal de ellas es la cantidad de estrellas que se pueden visualizar. El programador, Ronald Alpiar, ha optado por incluir estrellas de magnitud 4 o inferior (la "magnitud" es el nivel de brillo de una estrella, representando una magnitud alta a una estrella débil). A simple vista el ojo humano puede distinguir cuerpos celestes de magnitud 6, de modo que el programa no intenta mostrar todas las estrellas que podría ver una persona en una noche clara.

El libro que se incluye con el programa analiza los diferentes tipos de telescopios e informa de las mejores horas para observar los astros.

Luz de estrella, brillo de estrella
Las dos modalidades de visualización de *Starfinder* muestran un panorama del cielo rectangular o circular, según la línea de vista sea horizontal o vertical. La escena estelar se puede regular para un punto de vista situado en cualquier lugar de la superficie de la Tierra, en cualquier momento durante el s. xx. En la visualización se pueden buscar estrellas o planetas determinados, como el cometa Halley

Vista vertical

Vista horizontal

Starfinder: Para el BBC Micro y el Electron
Editado por: Century Software, Portland House, 12-13 Greek Street, London, W1V 5LE, Gran Bretaña
Autores: Heather Couper (el libro), Ronald Alpiar (el programa)
Palanca de mando: No se necesita
Formato: Cassette

mi computer mi computer mi computer mi computer

Editorial Delta, S.A.



10065

9 788485 822836